Artificial and Crowd Intelligence Based Recommender System Framework

Andreas Theodoros Lianos

First supervisor: Dr Linda Yang
Second supervisor: Dr David Ndzi
Third supervisor: Dr Branislav Vuksanovic

Department of Engineering
University of Portsmouth
United Kingdom
June 2016

The thesis is submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy of the University of Portsmouth

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.

Word count: 37825

ACKNOWLEDGEMENTS

First and foremost I would like to thank my family, my friends and my other half, for their practical and emotional support, their encouragement and their faith, and the sacrifices that they have done so I could complete this work. They experienced all the ups and downs of my research, and I hope that they now take the same pleasure in its completion as I do.

I would also like to express my gratitude to my supervisor Dr Linda Yang, who had the faith in me to provide this opportunity in the first place. Without her guidance and feedback, her encouragement and patience, but also the psychological support as a friend, this work would not have been possible.

Completing this work would have been all the more difficult were it not for the financial support of the University of Portsmouth, as well as for the proper provision of resources. A special thanks to Kostas Polyzos for his invaluable help and resourcefulness during the setup and execution of the experiments, and another special thanks to my current boss, Joe Cox, for his patience and understanding, if not his encouragement.

I would finally like to place on record my sense of gratitude, to all those who directly or indirectly, have lent their helping hand in this venture.

ABSTRACT

Tackling information overloading in online shopping is a significant challenge for e-shops. The current solutions demand that consumers become educated before they are able to distinguish which products are good for them. This research suggests a novel recommender system, namely ACIBa, which attempts to do part of the market research on behalf of the consumer and offer only a handful of products as recommendations.

The novelty of ACIBa is not just in its approach of limiting rather than widening the product alternatives, but also in that it bases its reasoning partially on artificial intelligence and partially on the collective intelligence of an arbitrary crowd of people. This work aimed to create a framework which provides the guidelines and basic tools to produce an ACIBa-like recommender system, as well as to create a live proof-of-concept system to showcase its use.

ACIBa made contributions in various areas. First and foremost it introduced a new recommender system that tackles the issue of information overloading from a very different perspective. The individual subcomponents of ACIBa have also extended the knowledge of their respective fields. The classifier used to cover the need for artificial intelligence, led to the creation of a novel ensemble classification methodology that allows training from typically unusable training sources. The software developed to interface between ACIBa and the crowd has been published as open-source software, effectively allowing other developers to create their own crowdsourcing systems. Finally, a new methodology has been introduced (namely ANA), that allows for an easy way to reduce the number of answers required from the crowd before conclusions can be reached.

Both these parts, i.e. the ensemble classifier and ANA, have undergone separate testing as standalone components. The ensemble classifier was shown to drastically outperform individual member classifiers. ANA has been found to consistently outperform the fixed number of answers distribution. An ACIBa based recommender system has been developed and demonstrated live to consumers. We used online questionnaires to gather feedback on the quality of results. Despite its infant stage, the response was generally positive and the system well received.

DISSEMINATION

Lianos, A. and Yang, Y. (2015) Classifying Unstructured Text Using Structured Training Instances and an Ensemble of Classifiers. *Journal of Intelligent Learning Systems and Applications*, **7**, 58-73. doi: 10.4236/jilsa.2015.72006.

CONTENTS

1 Introduction 1

1.1 Motivation 1

1.2 Aims and Objectives 5

1.3 Contributions 5

1.4 Thesis Outline 6

2 Background and Related Work 7

2.1 Recommender Systems: main approaches and challenges 7

2.2 Ensemble Classification 11

2.3 Crowdsourcing 16

2.3.1 Minimizing the required number of answers 17

3 Artificial and Crowd Intelligence Based RS Framework 20

3.1 System architecture 21

3.2 Web Crawler 23

3.3 Crowdsourcing Platform 25

3.4 Product Analyser 27

3.4.1 The Bullet Classifier 27

3.4.2 Attribute Merger 31

3.4.3 The Value Merger 34

3.4.4 Logic Controller 35

3.5 Recommendation Engine 39

3.6 Summary and Conclusions 44

4 The Ensemble Classifier 46

4.1 Introduction 46

4.2 Methodology 48

4.2.1 Introduction 48

4.2.2 Pre-Filtering of classes and training data 49

4.2.3 The member classifiers 51

4.2.3.1 The Name Classifier (NC) 52

4.2.3.2 The Value Classifier (VC) 53

4.2.3.3 Units Classifier (UC) 55

4.2.3.4 Special Words Classifier (SWC) 57

4.2.4 The ensemble 59

4.3 Evaluation 61
4.3.1 Set-up 61
4.3.2 Classification Accuracy 63
4.3.3 Method effectiveness 65
4.4 Summary and Conclusions 67

5 Adaptive Number of Answers (ANA) 69
5.1 Fixed number of answers 70
5.1.1 Methodology 70
5.1.2 Results and discussion 72
5.2 Adaptive number of answers (ANA) 74
5.2.1 Methodology 74
5.2.2 Results and Discussion 75
5.3 Summary and Conclusions 77

6 Demonstration and results 78
6.1 Testing the hybrid adaptive methodology 78
6.2 Application of ACIBa 80
6.3 Producing recommendations 86
6.4 Evaluation of MarketTroll from the users 88
6.5 Summary and Conclusions 98

7 Conclusions 99
7.1 Discussion and future work 99

References 103

Appendix A – Intermediate LC results 108
Appendix B – Product Recommendations 111
Appendix C – Questionnaire Sample 113
Appendix D – UPR16 Form 119

LIST OF FIGURES

Figure 2.1 Example of a typical product listing webpage 12
Figure 3.1 Overall architecture of ACIBa 22
Figure 3.2 HTML structure of a product webpage 23
Figure 3.3 The basic information flow of the Crowdsourcing Platform 25
Figure 3.4 Screenshot of MarketTroll 26
Figure 3.5 Snapshot from an actual webpage 28
Figure 3.6 Classification of attributes during merging with other attributes. 31
Figure 3.7 Example scoring histogram for the attribute "(sensor) resolution". 41
Figure 4.1 Example flow of the ensemble classification process. 49
Figure 4.2 Correct predictions of each classifier 64
Figure 5.1 Representation of an answer generator. 70
Figure 6.1 Distribution of difficulties 80
Figure 6.2 Screenshot from the results page served to the users 88
Figure 6.3 Results of Question 1 90
Figure 6.4 Results of Question 2 91
Figure 6.5 Results of Question 3 92
Figure 6.6 Results of Question 5 94
Figure 6.7 Results of Question 6 94
Figure 6.8 Results of Question 7 95
Figure 6.9 Aggregated answers to Question 8 96

LIST OF TABLES AND ALGORITHMS

Table 3.1 Example result of the Web Crawler 24
Table 3.2 Example transformation of score values 43
Table 4.1 Creation of training instances for the NC 53
Table 4.2 Creation of training instances for the VC 55
Table 4.3 Creation of training instances for the UC 56
Table 4.4 Example Cases where special character meaning is assumed 57
Table 4.5 Creation of training instances for the SWC 58
Table 4.6 Example Classification of a the bullet "24 Megapixels" 59
Table 4.7 Evaluation Datasets 61
Table 4.8 Training Data Gathered for the Camera Dataset 62
Table 4.9 Training Data Gathered for the TV Dataset 63
Table 4.10 Important Attributes Identified for the Camera dataset 66
Table 4.11 Important Attributes Identified for the TV dataset 66
Table 5.1 Benchmark of fixed number of answers distribution 73
Table 5.2 Benchmark of ANA 76
Table 6.1 Achieved accuracy for different stop conditions (c) 79
Table 6.2 Attributes identified as important by ACIBa. 81
Table 6.3 Merged attributes and their respective scores. 83
Table 6.4 Identified types for the important attributes 83
Table 6.5 Important attributes and their possible values 84
Table 6.6 Grouping of answers for Question 4 92
Table 6.7 Questionnaire platform bias test 97

Algorithm 4.1 Removal of rare classes 50
Algorithm 4.2 Outlier detection using Interquartile Range Test 50
Algorithm 4.3 Removal of long attributes 51
Algorithm 4.4 Selecting the best weights 60
Algorithm 5.1 Randomizing the distribution for a given number of options 72

ABBREVIATIONS

ACIBa ..... Artificial and Crowd Intelligence Based RSF

ANA ..... Adaptive Number of Answers

CF ..... Collaborative Filtering

CP ..... Crowdsourcing Platform

DOM ..... Document Object Model

LC ..... Logic Controller

MLP ..... Multilayer Perceptron

mTurk..... Amazon's Mechanical Turk

NC..... Name Classifier

PA..... Product Analyser

RE..... Recommendation Engine

RS ..... Recommender System

RSF..... Recommender System Framework

SWC..... Special Words Classifier

UC..... Units Classifier

UI..... User Interface

VC..... Value Classifier

1 INTRODUCTION

E-commerce is becoming increasingly popular as a way of purchasing retail goods. In a report published by the USA census bureau office, it was clearly shown that between 2002 and 2010 retail e-sales increased at an average annual rate of 17.9%, while the total retail sales had an increase rate of just 2.6% (U.S. Census Bureau, 2012). Online markets have allowed sellers to offer a greater product selection, however, offering more products does not necessarily imply that the customers will have the cognitive ability to evaluate them all as alternatives. On the contrary, there is strong research evidence that suggests that as the number of products increases beyond a point, information overloading takes over causing a series of negative effects to consumers.

Online markets have a range of tools in their arsenal to tackle this issue. However, none of them sufficiently targets uneducated consumers, that is consumers who do not know what to look for in a given product category. This research project addresses this gap and investigates the plausibility of a Recommender System (RS) that aims to limit the scope of available products, and thus reduce information overloading in online shopping for uneducated consumers. The goal of the system is to do the bulk of the market research automatically on behalf of the user, and recommend only a handful of products that represent good and relevant options. The system bases its reasoning partially on Artificial Intelligence, and partially on crowdsourcing, a novel approach to RS.

To examine the above, a proof-of-concept system was built and made publically available, utilising Compact Digital Cameras as an example for demonstration. Much of this work is devoted in the development of a methodology that allows the extraction of knowledge from an arbitrary crowd, with the help of artificial intelligence. The quality of the recommendations and the overall success of the demonstration system were evaluated via questionnaires.

1.1 Motivation

Information overloading refers to the inability of a system to process all the information that it has been given (Eppler & Mengis, 2004). Research has shown that information overloading triggers perplexity and restricts the capability to process the existing information or to perceive new one (Schick, Gordon, & Haka, 1990). Many researchers have linked information overload with one's wellbeing and satisfaction. In his book "The paradox of choice", Schwartz (2005) argues that even though having enough choices is necessary to our relative wellbeing, having too many of them works in the opposite direction. Similarly, Reutskaja and Hogarth (2009) defined satisfaction as the difference between the benefits and the cost of choice, and argued that after a point costs escalate faster than benefits. As a result,

satisfaction is an inverted U-shaped function of the alternatives considered, describing why having too many options is not always a good thing. In their experiments, Iyengar and Lepper (2000) provide a lot of compelling experimental evidence that having too many choices may sound more appealing, but it has detrimental consequences for human motivation and undermines the chooser's satisfaction. A possible explanation for this comes from Scammon (1977), who argued that consumers split their available time examining all the possible options, and with a limited amount of time, more options means less time spent in evaluating each.

Simplicity is yet another factor that may be important in decision making. In general, the simpler and clearer the attributes of a product or a service are, the more likely it is to be prefered by consumers, since it is likely to lift off the anxiety of making a wrong choice due to not completely understanding what they buy. Iyengar, Huberman, and Jiang (2004) for example reported that as the number of investing options within the offered pension plans increases, the percentage of employees who participate in any plan decreases. Mottola and Utkus (2003) estimated that number to be 2% for every 10 investment options. Another side of the complexity coin is the relative difference between available products, rather than their absolute "score". That means that consumers are more reluctant to make a choice when the available products are similar to each other, in comparison to buying a product that is clearly superior than the rest, even if the product itself is not that good (Dhar, 1997).

Source: http://www.gocomics.com/calvinandhobbes/1995/09/21/

All the above studies seem to suggest that the benefits of having too many options quickly diminish, giving place to a series of problems that negatively affect the final result. Whether this is an objectively worse choice or a less satisfying experience for the consumer, the phenomenon of information overloading in decision making needs to be addressed. Arbitrarily limiting the number of options in a market is not a viable solution, since the greater variety is more appealing to consumers. After all, removing products from only one market does not really limit the scope of options for the consumer. Hence, consumers and

markets alike need to find mechanisms to limit the number of options without limiting the number of products.

Online environments exacerbated the problem by allowing ever increasing product ranges. The UK amazon marketplace offers over 2500 cameras in the point-and-shoot category alone[1]. It is evident that evaluating all the product alternatives is not a realistic scenario. Lee and Lee (2004) reported findings that support that on-line information overload results in *"less satisfied, less confident, and more confused consumers"*. Interactive research aids have been developed to assist consumers limit their scope of options and make better comparisons (Häubl & Trifts, 2000). Degeratu, Rangaswamy and Jianan (2000) suggested that factual information has greater effect on online markets rather than on traditional stores, which further highlights the need for good comparison tools. In their work about the importance of user interface (UI) on e-shops, Lohse and Spiller (1998) also highlighted the importance of good search engines, saying that *"search engines should be mandatory for all large Web sites"* (Lohse & Spiller, 1998, p. 86). In a subsequent study, they found that big stores are not as effective as small stores in converting traffic into sales, because consumers may not find what they are looking for (Lohse & Spiller, 1999). They suggested that improved search functions and navigation links might overcome this disadvantage. Although their research is now over 15 years old, it is evident that their suggestions have been adopted by virtually every modern e-shop, further strengthening their point. Senecal and Nantela (2004) reported that consumers preferred products twice as often if these have been recommended by an online recommendation source, which highlights the important role of RS. Finally, Punj and Moore (2007) found that recommendation agents that can filter and integrate information, and offer feedback, influence the consumer's decision more, in comparison to agents that are only aware of the alternative options. This means that for the consumer, making a recommendation is not enough and a justification is also needed.

E-markets need to employ mechanisms to alleviate the problem and assist the decision making process. Among the most commonly suggested mechanisms in literature are comparison tools, good search engines and quality recommendation agents (Degeratu et al., 2000; Lohse & Spiller, 1999; Punj & Moore, 2007; Senecal & Nantel, 2004). Mechanisms that limit the scope of options are also used, such as category navigation and filtering. These methods work rather well for users that know what they are looking for, but this is not the

[1] Results occurred from personal research www.amazon.co.uk after filtering for products under "Electronics and Photo > Camera and Photo > Digital Cameras > Point & Shoot Digital Cameras". Then, in order to filter out misplaced products (such as cases or screen protectors), products are refined to only those that list any number of "Megapixels". The exact number is 119 pages of 24 products, plus 11 (2867). Last accessed 13 May 2015.

case for consumers who look into a product for the very first time. For example, a consumer looking to buy a camera for the first time would not know how many megapixels are reasonable, if 5x optical zoom is good enough or the importance of HDR. Consequently, the uneducated consumer is unable to effectively use filtering, searching or make comparisons; virtually everything in the toolkit provided to help with information overloading, requires education. Current approaches with RS do not address this issue either. They aim to help users discover more novel and relevant products, rather than limit their scope of options in a specific product search. This is likely the reason why RS have been widely adopted in intellectual goods such as books or movies.

Guided selling is a possible approach to the above problem, targeting users that have just been introduced to a category. It addresses the issue by converting user needs to specific product attributes. Usually found in the form of a step-by-step questionnaire, guided selling is in essence placing the filters on behalf of the user based on their answers. However, this technique is not without its own disadvantages. To create such a guide the relevant knowledge needs to be extracted from an expert, and as a result the process cannot be automated. In turn this makes it very expensive and thus scarce. To examine this, a short research on 10 well-known web sites was performed, where we find that none of them offer guided-selling, while two of them offer a "buyer's guide" that educates consumers on what to look for when shopping for a camera[2]. Guided selling might help restrict the product alternatives, but it does not completely address the issue. Finally, even after very tight filtering, the resulting amount of products can still be overwhelming, especially because following such a filtering the products are bound to be similar to each other.

To address the issue of information overloading effectively, one would require to know which product attribute relates to which user need (e.g. the snowboarder's height dictates the suitable length of the snowboard). It is the extraction of such knowledge that forms a significant challenge, and has thus far been an obstacle in automating guided selling techniques. If this knowledge was available, the user would only have to state his needs, which could then be translated to specific requirements, which in turn can be used to select products that meet those requirements to present back to the user.

1.2 Aims and Objectives

This work suggests a novel, Artificial and Crowd Intelligence Based (ACIBa) Recommender System Framework (RSF). ACIBa targets online markets and the uneducated consumer, and it is tailored to products where the choice is based more on objective criteria

[2] The top 10 sites offering DSLR sales were examined, as these occur from Alexa rankings http://www.alexa.com/topsites/category/Top/Shopping/Photography. Last accessed 08 Aug 2013.

(e.g. a TV) rather than subjective preference (e.g. clothes). It aims to offer a manageable number of options to the user, with meaningful trade-offs at the core product features, both of which have been shown to reduce information overloading.

ACIBa bases its recommendations on knowledge of which attributes are important for a particular use of the product, as well as on what values are required for those attributes. To continue on the previous example, ACIBa would need to learn that the length of the snowboard is an important attribute that greatly affects decision making, and that if the snowboard is to be used by a beginner that is a 170cm tall, a proper length would be ~152cm. Similar knowledge is found in guided selling systems, and is traditionally extracted from experts and encoded into the system by the system developers. In order to become automated, ACIBa replaces the expert with a newly rediscovered source of knowledge and intelligence, the crowd. In this context, the crowd is an online, arbitrary group of people, and crowdsourcing is the action of handing out a small task to be completed by that crowd, typically in the form of a question. As it will be later discussed, crowdsourcing has been widely successful in providing computers with human intelligence in an automated way.

Specifically, this research has the following two objectives.

1. Create an RSF which provides the guidelines and basic tools to produce a recommender system that meets the above description.
2. Create a proof-of-concept RS and demonstrate its use.

1.3 Contributions

First and foremost, ACIBa contributes to the field of Recommender Systems by introducing a novel approach that looks at the problem from a very different perspective. It does not aim to discover more relevant products, but rather to reduce the existing number of products, in an effort to make the market more accessible to consumers. In essence, ACIBa is a tool aiming to take the role of the selling advisors found in physical stores and transfer it to online markets. Using the crowd as a source of knowledge and intelligence is also a novelty of ACIBa. Even though other RS do use the crowd as a source of information (e.g. user reviews), ACIBa uses the crowd to slowly progress its reasoning, until it can go from raw data to product recommendations (Chapter 3).

ACIBa contributes to other research fields through its individual components. In the path to identify the important attributes, ACIBa analyses webpages using Artificial Intelligence (AI). Specifically, it uses a novel ensemble classification technique that does not require the traditional training data of supervised classification, as it can be trained from different data sources (Chapter 4).

A challenging aspect of any crowdsourcing system is how many answers need to be gathered for each question. The typical method used is a fixed number of answers for each question, which as it will be later discussed has many disadvantages. In this thesis, I propose an adaptive methodology that allows each question to gather a different number of answers based on its perceived difficulty, which creates benefits in either accuracy or the total number of required answers (Chapter 5). This is achieved through a simple yet effective metric to gauge the difficulty of a question based on the already collected answers.

Finally, the Crowdsourcing Platform that creates the link between the application and the crowd is published as a standalone java library for other developers to use. It allows developers to create and submit questions to a central controller, as well as use a web based front end that can retrieve questions from the controller and serve them to the users.

1.4 Thesis Outline

This thesis is divided in 7 chapters, including this introductory section. The next chapter (Chapter 2) is a thorough review of the literature that examines relevant studies both for ACIBa as a whole and for its crucial subcomponents. It serves mainly to set the scene and introduce the reader to the topic area, but it also offers the motivation or rationale behind various design decisions. Chapter 3 describes ACIBa and its components in great detail, constituting the core of this work. Chapter 4 expands on the proposed ensemble classification methodology, also containing a standalone evaluation. Chapter 5 discusses the methodology for the adaptive number of answers (ANA) and compares it to fixed number of answers. Chapter 6 follows through the implementation of the actual recommender system, namely MarketTroll, and its intermediate steps until the generation of recommendations, followed by the results of the user's feedback survey. Finally, Chapter 7 briefly summarises the work presented in this thesis, along with important concluding remarks, and the many directions for future work.

2 BACKGROUND AND RELATED WORK

This chapter provides the background of the thesis topic area along with the relevant literature. Section 2.1 presents the main RS approaches, as well as why and under which conditions the existing solutions do not adequately address the issue of information overloading. Section 2.2 refers to the first major subcomponent of the proposed recommender system, the ensemble classifier. It helps to position the research in the world of classification by discussing similar research, why it might not be applicable and what we can learn from it. Finally, section 2.3 introduces the world of crowdsourcing, and subsection 2.3.1 examines a specific crowdsourcing challenge and some existing approaches, as well as it introduces a novel approach that comes from a different angle.

2.1 Recommender Systems: main approaches and challenges

Researchers have identified the issue of information overloading in e-shops and various solutions have been suggested to help resolve it. However, even with all the tools that have become available over the years the problem has not been sufficiently addressed in all contexts. Users who know what to look for and are willing to devote the time, can make use of the excellent filtering capabilities. Intellectual goods, such as books and movies, reap the most benefits from recommender systems, which do an excellent job in suggesting more relevant content. However, when an uneducated consumer attempts to buy a one-off product, there is little help readily available on what would be a good purchase.

Recommender systems (RS) have been suggested as an approach to the problem of the overwhelming market (Schafer, Konstan, & Riedl, 2001). They enjoy success in a wide range of applications, such as advertising, search engine results, and e-commerce (Agichtein, Brill, & Dumais, 2006; Joachims, 2002; Kazienko & Adamski, 2007; Mobasher, Cooley, & Srivastava, 2000). To produce their recommendations, such systems might use a wide range of criteria, often combined in hybrid systems. Possibly the most well-known approach is collaborative filtering (CF), where people are clustered into groups of 'similar interests'. The principle behind CF is that if users A, B and C all liked products 1 and 2, then if user D likes product 1, he is also likely to like product 2. For example, if a lot of people who liked "Lord of the Rings" also liked the "Pirates of the Caribbean", then if a new user liked the first he is also likely to enjoy the second, which can thus serve as a recommendation. CF is not limited to clustering similar users, in truth quite often the clusters are created around similar items, the principle however remains the same (Sarwar, Karypis, Konstan, & Riedl, 2001).

Another heavily used approach is content-based filtering. In this case recommendations work by matching the user's preferences to some aspect of content. For example, if a consumer bought a phone, TV and speakers from BrandA, then they are likely to like BrandA,

so BrandA products will be recommended to them. Of course this is an oversimplification of the technique, but it is beyond the scope of this thesis to present a detailed analysis of the existing recommender systems. A comprehensive summary and comparison of the most common recommendation techniques can be found in the excellent work of Burke (2007), while the Recommender Systems Handbook (Ricci, Rokach, Shapira, & Kantor, 2011) provides more in-depth information. This work emphasises as to why recommender systems in their current form, cannot sufficiently address the problem of information overloading in all contexts.

In a panel interview held during RecSys 2010, Palash Nandy, manager of YouTube Discovery, said that what he regarded as the best RS application to date was built for music. He stated that the main reason for his choice was that *"users are interested in discovering novel music"* (Guy et al., 2010). In examining the use (and misuse) of RS, one needs to consider their link with novelty. It can be argued that novelty is inherently connected to all intellectual goods such as books or movies; however, users are not necessarily interested in discovering "novel" cameras but rather just in taking a good purchase decision. Recommender systems however have often been misused, as the same methodologies are applied despite this clear distinction between products. A common scenario of obvious misuse is a returning customer to an e-shop that after a successful purchase still gets recommended products similar to the product they just bought (Sundaresan, 2011). This can be undoubtedly useful in the case of a book but not at all useful in the case of "non-consumable" items, such as phones or cameras. This distinction between consumable and non-consumable goods in the greater sense, may be the reason why there are so many standalone recommender systems, all built strictly around intellectual goods (e.g. www.jinni.com, www.whatshouldireadnext.com, www.tastekid.com, www.goodreads.com, etc.).

Recommender systems face a series of challenges that further hinder their success as decision making tools. The possibility of "biased" suggestions lowers the user's trust, making them a less favourable solution for helping users decide. Benbasat and Wang (2005) found that consumers may be concerned about the benevolence of the recommender agent, and its intent to act in their own best interest rather in the interest of the seller. The agent's competence in performing the task has also been questioned. Additionally, Senecal and Nantela (2004) found that user's trust in a recommendation is even more diminished to commercially linked sites, as opposed to non-commercially linked, third party sites.

Another challenge is that RS that personalize results based on the user's profile have an inherent problem dealing with new users, as they do not have any previous knowledge for them (Good et al., 1999; Rashid et al., 2002). This is particularly evident to consumers that

wish to buy "one-off", non-consumable products, which are often bought one at a time. Unlike books, or music, rarely will a customer return after a week to buy another, different cooker. But users newly introduced to the system are not the only problem. Originating from super-market basket recommendations, Burke (1999) introduced the “banana problem”, where very popular items—like bananas—get recommended along with everything. In the same research, Burke also suggested that for products bought less frequently and one at a time, traditional recommendation techniques cannot provide appropriate results.

In his book about recommender systems, Setten (2005) reports that such methods are only part of the solution, and goes as far as to suggest that *“they can sometimes, even increase the feeling of information overload”* (Van Setten, 2005, p. 5). It is not difficult to see why; from the point of view of an "overloaded" consumer who is looking to buy a TV, recommending more similar TVs to the one currently under examination only increases the number of alternatives that need to be examined. In addition, as it was previously discussed, offering similar products makes the decision even more difficult, and RS methodologies do not always attempt to diversify their results. In essence, recommender systems are unable to actually limit the scope of options, something that is mandatory in reducing information overloading.

Recommender systems are not badly designed, they are just a solution to a different problem. The problem they address is not to assist confused consumers in making a decision, but rather to help consumers discover relevant products in a product-flooded market. This is why despite their wide adoption, there is an inability to specifically assist users that are looking into a category for the first time. In an executive technology report from IBM, Robyn Schwartz identified this exact scenario when he was asked for an example of usage for guided selling; *“a user who is new to a product”* (Andrews, 2005). His view was that shoppers who do not know much about the product category will require education on what to look for and how to shop. A white paper published by excentos, a company that specialises in guided selling solutions, described the above situation with an example, that of a beginner user attempting to buy a snowboard (Tangermann & Streit, 2011). The paper explained that all the alternative methods (i.e. free-text search, category browsing, filtering and recommendations) fail, because none of them is capable to translate the only knowledge the user has—his height—to a requirement about the suitable length of their snowboard. Guided selling (a category of knowledge-based recommender systems) was finally proposed as the only method capable to assist the user in making an educated decision.

Guided selling is a very useful method that has multiple successful applications on today’s market and marketing. For example, devices with preloaded guided selling software are used in the point-of-sales to assist customers with complex buying decisions (‘Guided Selling’,

2005). It is a mechanism developed to fill the gap of search engines and filters for the users who are newly introduced to a category and do not know what filters to apply (Hawley, 2012). In essence, guided selling takes the role of sale assistants found in physical stores, and transfers it to e-markets.

For example IKEA offers guided selling for new mattresses. Among other questions, the guide asks whether the customer sleeps on his back, his side or his stomach, in order to determine and propose the hardness of the mattress[3]. Another good example is the guided selling offered for cameras on Pixmania. Among other things the customer is asked if the intended usage is for "everyday use" or "artistic or professional photography", so the system can decide between compact and DSLR cameras[4]. Both examples demonstrate how guided selling can transform the user's knowledge to explicit needs, without requiring the user to become educated about the product category.

Knowledge-based recommender systems operate on the principle that information about the customer's needs must be acquired before making a recommendation (Burke & others, 1999). The system can then take a knowledge-based approach to generating a recommendation, by finding what products meet the customer's requirements. As Burke et al. tell us, the motivation for this approach is that existing recommender systems do not help users identify the product they need, and recommendation techniques such as collaborative filtering are not effective for products bought less frequently and one at a time. The situation does not seem to have changed since Burke made that observation; yes recommender systems have significantly improved, but they have not changed direction.

Knowledge-based recommender systems however require that specific domain knowledge is built into them. In Burke's words, they also require *"knowledge engineering with all of its attendant difficulties. For a system to make good recommendations, it must understand what features of products matter. It must have access to a product database in which those features are readily discernible or at least inferable"* (Burke & others, 1999). This is a major drawback as such systems require major effort to maintain, especially in today's market where the products that are offered change daily. Another challenge easy to assume, is that after the system has identified all the products that meet the customer's requirements, the resulting amount might still be too much to be handled by the user. In addition to quantity, such products are also highly likely to be similar to each other. This amplifies the problem and further hinders the ability to reduce information overloading, since previous findings

[3] As seen at http://www.ikea.com/gb/en/about_ikea/bedroom_guides.html. Last accessed 12 Jan 2013.

[4] As seen at http://www.pixmania.co.uk/buyers_guides-g.html. Last accessed 12 Jan 2013.

have suggested that it is more difficult to choose between similar products. With these major drawbacks in the system's effectiveness and a diminished user's trust, it becomes reasonable that they have not enjoyed much commercial success.

Summarising, we initially identified the problem of information overloading, and saw how too many product alternatives can have adverse effects for the consumer and the market alike. Markets have a series of tools in their arsenal to help consumers discover products relevant to them, each with their merits and challenges and each serving different customer needs. The ubiquitous passive methods such as search and filtering are only useful for consumers that know what to look for. Active methods such as recommender systems are not applicable for non-consumable goods that are often one-off purchases, as they aim to help consumers discover more relevant products rather than reduce information overloading. Finally, guided selling has the potential to address this issue, but it is very costly to develop, and it comes with several disadvantages that can hinder adoption and limit success.

Based on the existing literature, an optimal system should be one that is able to examine many alternatives and select only a few suitable products for recommendation. The final products should be diverse, with each product bringing something different to the table. The user should then be able to easily select between the meaningful trade-offs of the recommended products. The system should offer a justification of its results, explaining why all the other products are not good enough, as well as offering easy comparison points between the recommendations. Finally, the system should be able to operate outside any commercial sites, offering advice about the product, rather than the marketplace it should be bought from.

2.2 Ensemble Classification

To build a recommender system with the above characteristics, one of the major challenges Burke (1999) has identified needs to be addressed; *"For a system to make good recommendations, it must understand what features of products matter".* To discover such features, this research attempts to identify those that are being consistently highlighted by e-shops. This is based on the premise that features consistently highlighted for many products are the ones that shape the decision making process. This information is obtained by utilising the short bullet-point texts (for short, bullets) that precede the product's extended description and highlight its salient attributes. An example of such bullets can be seen in Figure 2.1.

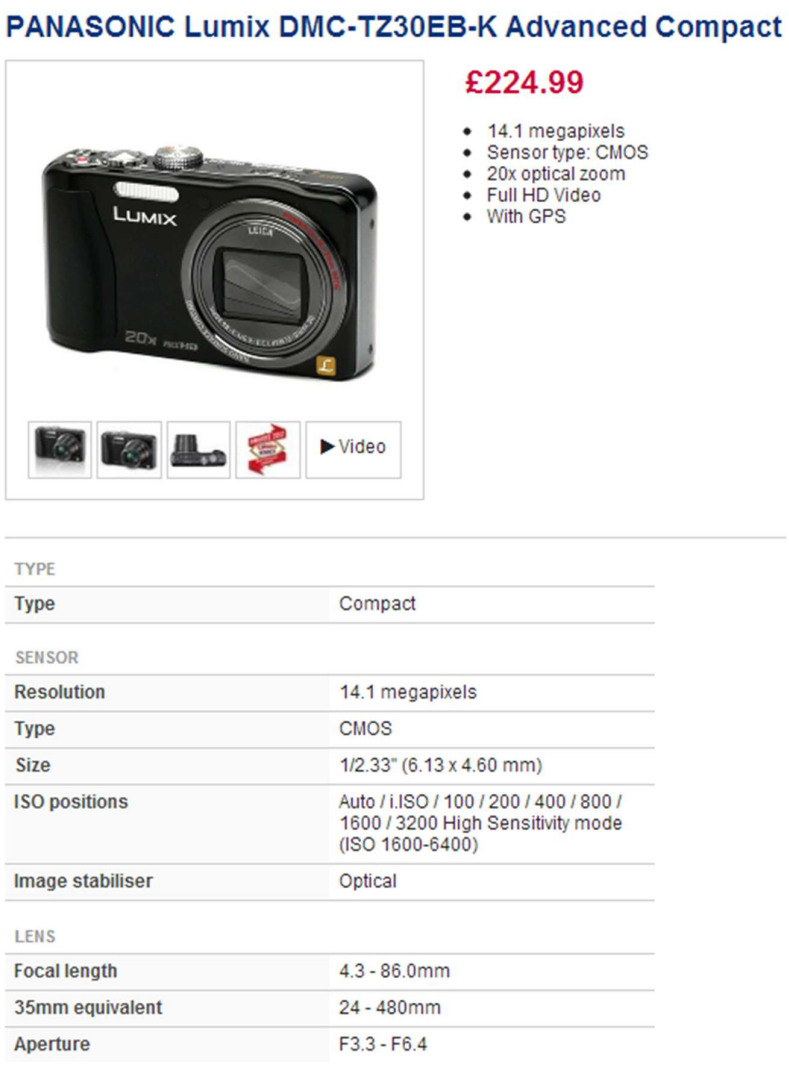


Figure 2.1. Example of a typical product listing webpage.

This can be a very challenging task to automate, because bullets are unstructured text and identifying to which attribute(s) each of them may refer to can be difficult. The problem of text classification is not new and there is ample literature relating to it (Forman, 2003; Manning, Raghavan, & Schütze, 2008; Rennie, 2001; Sebastiani, 2002). The most relevant application comes from information retrieval where documents are ranked according to their similarity to a given query. Transferring the technique, the attribute's names can be ranked according to their similarity with the given bullet. For example the bullet "24x optical zoom" is very similar to the attribute's name "optical zoom", so that would probably rank first. The main limitation when applying this technique is that a bullet may or may not contain the same words with the attribute's name. For example, the bullet "24 Megapixels" has no similarities with its respective attribute name "Resolution". As a result looking only at the

attribute's name makes the correlation between them impossible, rendering a simple application of text classification methodologies ineffective.

To overcome the previous limitation it may be useful to borrow from a methodology used to match the fields of one database to those of another (Dasu, Johnson, Muthukrishnan, & Shkapenyuk, 2002; Naumann, Ho, Tian, Haas, & Megiddo, 2002). This method examines the contents of the database (rather than the table names) to look for columns whose contents look similar. This however causes the opposite problem of the one previously mentioned, favoring the bullets that contain values over those that contain names. For example the bullet "24 Megapixels" might not match the name of the attribute (resolution), but it will match other similar values, such as "12 Megapixels", "18.3MP", "21.0 Mega Pixels" and so on. The bullet "Waterproof" however, has no similarities with the respective values of the attribute which tend to be either "yes" or "no", so the correlation is impossible. In addition, different attributes may have similar values; for example, height and width may both be 10cm, and waterproof and shockproof may both be either yes or no. This allows for false positives to occur, and is considered a major limitation to the approach (McCallum, Nigam, & others, 1998).

In general, the reason why both of these methods have limited applicability is because one cannot be sure which information is or is not contained in a bullet. At the same time there are no training instances for bullets, so that information cannot be learned either. This means that without manually classifying bullets to be used as training instances, traditional classification techniques cannot be used. However, the detailed table of product attributes and their respective values is nearly always available. Each of the above techniques can be used by using those tables as training data, even though they will be effective only some of the time. The two classification approaches can be merged into one ensemble classifier, lifting most of their limitations while benefiting from their strengths. Each classifier of the ensemble can be trained to look for different criteria, better utilising the available information.

Ensemble classification is a well-established research area with reliable results. It strives to replace a single classifier with multiple member classifiers whose results are then recombined. The member classifiers of an ensemble are similar, but not the same. The main diversification points of the various ensemble methods are 1) how to obtain the member classifiers, and 2) how to combine their results.

Despite their old age, Bagging (Breiman, 1996), Boosting (Freund, Schapire, & others, 1996; Schapire, 1990) and their later variations, are still the dominant ensemble classification methodologies. Maclin and Opitz (2011) have rather recently shown that they still outperform single classifiers. These methodologies meet at a common point; the member

classifiers are obtained by diversifying one base classifier, and as a result all members are similar to each other. The proposed ensemble, requires each member to have the ability to look for completely different characteristics in a bullet, and as a result the feature sets (i.e. the criteria the classifier uses to make a judgment) of each member classifier need to be different. Nonetheless, there is much to learn from the relevant literature.

Hansen and Salamon (1990) have shown that for an ensemble to be successful, member classifiers should be, among other things, diverse. "Diverse" in this context, means that given the same value to classify each member should be making different errors. Dietterich (2000) has later explained why this requirement is true. In simple terms, if members are not diverse, then when one is mistaken, all will. On the contrary, when the classifiers are diverse, when one is mistaken the others might be correct. In the proposed ensemble, the member classifiers have completely distinct feature sets that do not overlap. In truth the classifiers are designed so that where one fails, the other excels, maximizing their diversity.

Having identified the way to obtain the member classifiers, the interest shifts to the second point; the way that the results from members are combined. The prevailing approaches are plurality voting (Hansen & Salamon, 1990) and simple averaging depending on the type of task (Opitz & Shavlik, 1996). Simple averaging was used in the initial implementation of bagging (Breiman, 1996), but various studies have suggested that weighted averaging might improve accuracy (Perrone & Cooper, 1992; Rogova, 1994). Most weighted average methodologies, however, either assume that the classifiers are similar, or that it is possible to calculate a measure of the effectiveness of the classifier by using part of the training data. None of these assumptions hold true for this work. In a nutshell, this happens because the classifiers are trained from the structured table of attributes, but then classify unstructured text from the bullet-points. Since there is no prior knowledge of where bullet-points should be classified, the effectiveness of the classifier cannot be assessed without hand-tagging data. At the same time, creating evaluation instances from the training data is guaranteed to suffer from overfitting. These points will be better illustrated in Sections 4.2 and 4.3, which discuss the methodology and the evaluation of the proposed ensemble classification approach.

That is not to say that existing methodologies have nothing to offer. Following one of the most successful approaches, the developed ensemble classifier uses the weighted average to combine the results from the member classifiers. Since the weights cannot be calculated automatically from the data, a fixed weighted scheme has been used instead. This is possible because the classifiers themselves are not created dynamically and as a result the relation between them can be predetermined. Another approach suggests the use of the perceived

confidence of the classifier as a start for calculating weights (Maclin, Shavlik, & others, 1995). This is an angle worth exploring further as the approach is potentially applicable. However, because classifiers are different, and are not all implemented by the same classification methodology (e.g. Neural Networks, Naive Bayesian, Decision Trees, etc.), a comparison of their confidence levels might be complicated. Even if this methodology is not directly applicable, the core idea can be transferred and used. As a result, the weight of any classifier might be set to zero when its opinion is deemed not relevant.

A lot of research has been done on the classification techniques themselves. Simple Bayesian classifiers have been found to perform very well, especially with small feature sets (Friedman, Geiger, & Goldszmidt, 1997; McCallum et al., 1998). Another popular approach to classification is Neural Networks. As Neural Networks attempt to find connections between features, they are particularly effective where features are strongly correlated. They are however expensive to train, especially as the number of features grow (Ruck, Rogers, & Kabrisky, 1990; Zhang, 2000). Finally, Decision Trees is another widely used classification approach that has been considered for this work (Quinlan, 1986; Rokach & Maimon, 2008). Even though not used in any of the final classifiers of our proposed methodology, one should certainly include them in the possible pool of options if transferring the technique to a different context. An extended literature review of classification techniques is out of scope of this work, but it is worth mentioning that Kotsiantis et al. (2007) have written a thorough review of the most common classification techniques to date and Xhemali et al. (2009) a comparison of Naïve Bays, Neural Networks and Decision Trees.

Reflecting on the related literature, we see that even though this research draws upon the ideas and findings of the existing literature, it cannot be directly compared with any of them, mainly due to the member classifiers having different feature sets from each other. This work examines how to tackle the problem of training data being in a different format from the values that they will later classify. Partially addressing this problem by creating multiple but different classifiers, we also examine if these can be used effectively in an ensemble. There seems to be little knowledge in the existing literature on how to address both these issues. Concerning the development of feature sets, no literature could be found specifically tailored for consumer electronics. Naumann et al. (2002) discusses an application that matches fields of one database to those of another. The proposed feature set is rather generic, and as such has been used as a starting point for value-based matching.

2.3 Crowdsourcing

Identifying the attributes that matter most is only a small part of the solution. Before those attributes can be used to compare products and make recommendations, domain specific knowledge needs to be acquired. That is the key component of a knowledge-based recommender system, and the one that allows it to bind the user's needs to specific product requirements. In typical systems that knowledge is extracted from experts, which is also one of the limitations as it significantly raises costs. To overcome this limitation, we can turn to a different source of knowledge, people.

Crowdsourcing *"represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call"* (Howe, 2006b). In his relevant article in WIRED magazine, Howe (2006a), who introduced the term, offered numerous examples of companies that had already changed their strategy and turned to this newly discovered source of brain power for their operations. The term has since evolved to include tasks that are solved by multiple non-experts, who only collectively can provide a solution. Maybe the most predominant example is wikipedia, where over 100.000 people contribute monthly to sustain the world's largest encyclopaedia. Another wildly successful project based on crowdsourcing is Google maps. With the introduction of Map Maker, Google has effectively allowed anyone to edit and update maps. As Katragadda (2008) characteristically stated in the blog post unveiling the Map Maker, *"Building a map is an exceedingly complex process, and it is a challenge to reach all parts of the world"*; a problem that Google solved with the help of crowdsourcing. Google has since taken that a step further, with the subsequent crowdsourcing of street view imaging, and by acquiring and incorporating Waze, a GPS that offers real-time, user-generated traffic data. The world is now full of examples, Quirky is a platform where people collectively research and design new innovative products, the CAPTCHA project utilises millions of people to do Optical Character Recognition, and Galaxy Zoo is a project where people classify galaxies from space images. The list is endless.

Going a step further than a crowdsourcing project, Amazon's Mechanical Turk (mTurk) is a crowdsourcing platform that allows easy access to a crowd for any project that requires it. Essentially it is a marketplace for work, where developers can programmatically (through an API) ask questions that require human intelligence to be answered in natural language. Humans, or workers as they are typically termed, can then pick up the questions and answer them for a small monetary reward. The answer goes back to the programmer through the API, effectively allowing computer programs to gain access to human intelligence, and at very low cost (Barr & Cabrera, 2006).

2.3.1 Minimizing the required number of answers

Systems that require answers from multiple workers for a single task, often operate with an empirically determined, fixed number of answers per question. For example, Franklin et al. (2011) suggests an odd number to enable majority voting for quality assurance, typically 3 or 5. For their evaluation, Bacharach et al. (2012) use questions with 8 answers. In their study using mTurk, Kittur, Chi & Suh (2008) use 15 and 20 answers respectively for different tests. Interestingly, the underlying platform of zooniverse responsible for selecting the next subject to classify, cellect[5], will attempt to even out the number of responses each image has received. We can thus assume that when Lintott et al. (2008) stated *"~38 classifications per galaxy"*, the selection mechanism was trying to even out the responses to each question around that number. In all the aforementioned studies the selection of the limit is arbitrary and no explanation is given, depending mostly on the resources each system has available. Even though the fixed-number approach is common, the implications of different numbers are not thoroughly examined.

That is not to say that the fixed number approach is the only one, but rather that is commonly used in both research and industry. Numerous researchers have studied ways to either increase accuracy or reduce the number of collected responses. This is often done through estimations about the quality of the worker or the uncertainty of the results.

Liu et al. (2012) have suggested such an approach to deciding how many answers are needed per question. They proposed a formula that can be used to calculate this number for each individual question, based on the historical accuracy of workers and the expected accuracy of the results. This methodology implies that the historical worker accuracy is known or can be inferred, which might not always be the case. If each worker only answers one or a few questions, the ability to estimate their effectiveness is reduced or made completely irrelevant. In addition, any such methodology will suffer from the cold-start problem since no information is known for new workers, something that is exacerbated at the launch of a new system when all workers are unknown. Finally, the accuracy of each worker depends on the difficulty of the question, which is subjective for different workers as these have different strengths and weaknesses. This results in workers performing differently for different tasks, potentially making their accuracy inconsistent across different types of questions. Gold standard questions are suggested to overcome most of these issues, but this still does not address the variance in quality that occurs not due to the worker's ability, but due to the question's difficulty.

[5]https://github.com/zooniverse/Cellect

Ipeirotis, Provost and Wang (2010) have suggested a methodology where the bias and the error rate of each worker can be separated, thus improving its perceived quality. They also introduce a solution to the cold-start problem by calculating estimations of quality when there are not yet enough data. Their solution does address many of the issues with worker quality ratings, though they still do not take into account the inherent difficulty of each question as a parameter that affects results.

Sheng et al. (2008) examine the results of obtaining many labels for training examples for classifiers through crowdsourcing. They report that even a straightforward, round-robin of obtaining extra labels for all examples has significant benefits over single-labelling, even if the labels are noisy. This finding is expected, yet very important, as it strengthens the approach of acquiring more than one answers per task. They then continue to use the estimated uncertainty of the acquired labels, to decide which examples are the best candidates for extra labelling. This optimization is very important, as it can increase the accuracy of the results by concentrating effort where it is really needed. The technique can be potentially transferred to other crowdsourcing scenarios since in essence the acquired answers do not have to be used for labelling. In their experiments however, they only deal with binomial questions where the labeller's quality is assumed to be fixed. They also assume that all examples have the same degree of difficulty, something that is not usually the case for real life scenarios. In their own words, Sheng et al. report that *"In our analyses we also assumed that the difficulty of labeling an example is constant across examples. In reality, some examples are more difficult to label than others and building a selective repeated-labeling framework that explicitly acknowledges this, and directs resources to more difficult examples, is an important direction for future work"*.

Castano et al. (2016) present a methodological approach based on consensus and trustworthiness techniques. They work on the premise that the more workers agree on a certain answer, the higher the chances that it is correct. They also define trustworthiness techniques to measure the worker reliability. They then use this measure of trustworthiness to tailor the composition of a group of workers, by involving those that are more able to express a sharable answer where needed. The approach requires answers to reach a certain level of consensus. If that cannot be reached with the initial set of answers, more answers are requested in a second round where the group has been selected to meet certain trustworthiness criteria.

As a final note it is important to clarify that worker quality methodologies are often applied in already collected data to improve the accuracy of results, e.g. by the removal of

outliers or by weighting the answers according to the worker's competence (as this can be estimated a-posteriori). For clarity it should be noted that, even though possibly intertwined, this is a different step that affects how the decision is made on which is the correct answer from the already collected answers, rather than on deciding a-priori how many answers need to be collected.

3 ARTIFICIAL AND CROWD INTELLIGENCE BASED RECOMMENDER SYSTEM FRAMEWORK

E-markets employ several mechanisms to assist consumers find what they are looking for. The commonly available tools, such as the advanced search capabilities and detailed filtering options, can help consumers who know what they are looking for to quickly navigate to the relevant products. Sophisticated recommender systems can propose relevant items based on a wide range of factors. They have proven very effective in helping consumers discover more relevant products, which is particularly important for intellectual goods. However, in order to sufficiently address information overloading, discovering the relevant products is not enough. What is required is a way to actively assist in the decision making process, by actually suggesting which products constitute good choices, while at the same time limiting these suggestions to a manageable number. Especially when it comes to one-off buys and non-educated consumers (which is a natural combination), most of the existing systems are rendered ineffective. Knowledge-based recommender systems is a potential answer to this issue, as it can translate customer needs into explicit product requirements. However, such systems come bound with their own disadvantages, especially a major difficulty in their automation. This stems mainly from the requirement of an expert, from whom the knowledge must be extracted.

This chapter presents an Artificial and Crowd Intelligence Based (ACIBa) Recommender System Framework (RSF). ACIBa is a modular framework for the creation of knowledge-based Recommender Systems, which utilise crowdsourcing as their source of knowledge. The motivation behind this approach is to overcome the limitation experts impose in the automation and maintenance of such systems. The proposed architecture makes use of AI to obtain the required information when this is possible, in an effort to minimize the tasks required by the crowd. The two are often interwoven, with a module of artificial intelligence doing the bulk of the work, and the crowd validating or fine-tuning the results. ACIBa tackles the problem of information overloading by actually limiting the available products the consumer has to examine as alternatives.

3.1 System architecture

ACIBa is tailored for cases where there is a need to identify the best products for a particular use. Within its scope, the definition of what is best, as well as the possible uses, are defined by the crowd. This makes the framework very flexible, since it can accommodate any product category that fits one basic criteria—having 'objectively' best products. The distinction might not be immediately obvious, but often the 'best' product is something completely subjective, such as when buying books or plates. Under those conditions making a generalised suggestion is unfounded, but the situation is very different when buying a TV or a bread maker. One can find specific features to quantify, for example the viewing distance broadly defines the best size for a TV, and its intended use defines the requirement for smart software or not, the Hz the TV should operate at, as well as the number and kind of connection interfaces. Looks tend to come second; rarely does someone walk in a store and picks the prettier TV without consideration of most important attributes first. Of course a purchase is never completely objective and personal taste is always a factor, but it can be argued that in general lines, there are product categories where consumers will favour objective criteria more than subjective, arbitrary preference. ACIBa is designed with these product categories in mind.

Figure 3.1 outlines the major components of ACIBa and how they interact. In brief, the web crawler scans product webpages and extracts the useful information including the product's attributes. The gathered data are analysed step by step by the Product Analyser (PA) through a collaboration of AI modules and the crowd. The Crowdsourcing Platform (CP) is an online application where the system can submit questions in natural language. CP mediates between PA and the crowd, by serving questions to workers, gathering the answers and passing back the results. The result of the Product Analyser is knowledge of which attributes are important to decision making along with the recommended values for these attributes. The Recommendation Engine uses this knowledge to score and rank the product database, with the top scoring products serving as recommendations. The Product Database also contains other important information, i.e. price, rating and number of reviews.

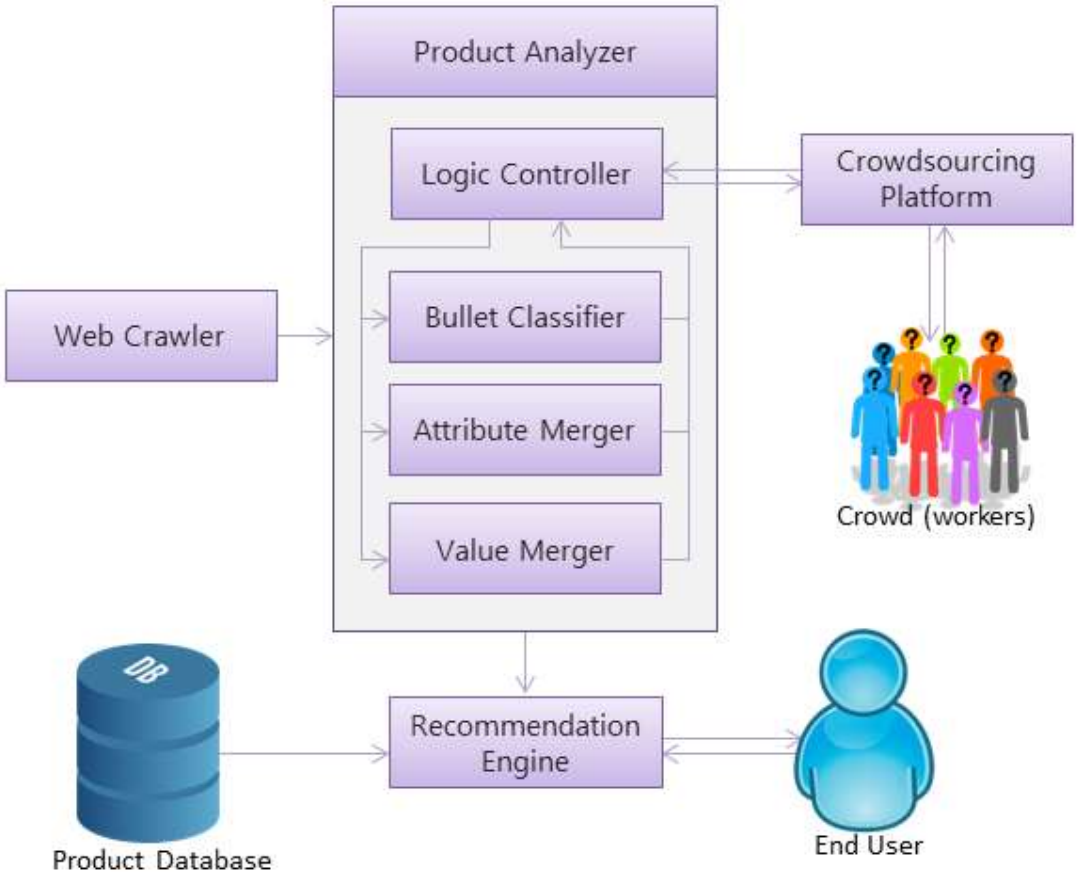


Figure 3.1. Overall architecture of ACIBa.

The modularity of the framework allows components to be swapped with virtually no modifications to other parts of the system. For example the Web Crawler functionality could be swapped with a pre-existing database, as long as that DB would contain all the required information. This also enables the independent modification and extension of each component. For example, CP is responsible for accepting questions and serving them to the crowd. The component could be swapped for an online work marketplace like mTurk, or it could be expanded to use different crowds for different reasons. Similarly some of the work that now requires the crowd's intelligence could be handled with artificial intelligence in the future. The modular design of the framework, allows the expansion of PA to include more AI sub-components, without affecting other parts of the system. The following sections discuss each component in greater detail.

3.2 Web Crawler

The Web Crawler is responsible for two distinct tasks, to acquire the relevant product webpages and to extract from them the required information. Both those tasks utilize the Document Object Model (DOM) that is used to describe all the elements of a webpage. Through it, the crawler can target specific objects in a webpage that might represent links or other important information. For example, the bullet point list of the product's description is represented in the DOM by a list object. Even though different web-sites have different structure, that table is distinguishable in the DOM by a unique combination of characteristics, such as its name, position, or styling rules. Figure 3.2 illustrates this example. The bullet point list is uniquely identified as "a descendant of a `DIV` with the class `product-highlight`, of type `UL` and with the itemprop `description`".

Figure 3.2. HTML structure of a product webpage. The DOM characteristics of the highlighted elements, uniquely identify the bullet point list.

To acquire the webpages the crawler is seeded with the URL of the product-listing webpage where all products are listed (possibly grouped in pages rather than all at once). The crawler is also given the unique DOM identification characteristics of two elements, the link to the individual product webpages, and the link to the next page of products. With this information available, the crawler is in the position to acquire each product's individual

webpage, and then move continuously to the next page of product listings until all products have been scanned.

A sub-module of the crawler, the content extractor, uses the same approach to identify the required parts of the product webpage. What is needed in this case is the unique DOM identification characteristic of the area that contains the required text. Four areas are extracted from each webpage; the bullet-point descriptions that typically accompany the product's image, and the names, values and headings of the attributes, from the detailed list of attributes.

This approach makes both the crawler and the content extractor very flexible, since they can be customized for virtually any web-site where the product webpages follow a structural pattern. The content extractor can also be expanded to identify any other important areas of information that might be wanted in the future. Furthermore both components are virtually infallible. These benefits come at the cost of scalability, since for each new market that needs to be scanned a new set of DOM characteristics needs to be identified. The required flexibility is also the reason why generic extraction frameworks have not been used. Specifically, Ghani et al. (2006) have developed a methodology that can extract pairs of attributes-values from webpages. However, due to the requirement for additional information (i.e. bullets and category headings) their work can only be half applied, and the same stands for other, more generic extraction frameworks.

The outcome of the web-crawler is product names, accompanied by a detailed table of attributes that includes names, values and headings, as well as any possible bullet point descriptions. Table 3.1 shows the outcome of the web crawler for the product shown in Figure 3.5, p.28. Similar data tables are created for all scanned products.

Table 3.1. Example result of the Web Crawler. Attribute headers are presented in brackets.

Name	NIKON D1500 DSLR…	
Attribute	(overview) Camera Type	Digital SLR
Attribute	(overview) Processor	Expeed 3
Attribute	(sensor) Resolution	24.4 megapixels
…		
Bullet	24.4 megapixels	
Bullet	Sensor size and type: 13.2 x 15.4 mm CMOS	
…		

3.3 Crowdsourcing Platform

The role of the Crowdsourcing Platform (CP) is to connect the rest of the system with the crowd, and in essence to offer access to human intelligence. It is able to accept questions in natural language, serve them to the crowd and collect the responses. CP is developed as a Java and JavaScript library for the backend and frontend respectively. The code is publically available as open source software at http://alianos.website/r?c=cplib. Other components of the system can communicate with CP through an API. For the backend there are two main interactions with the CP Controller, submitting a question and being notified when an answer has become available, while for the frontend the interactions are requesting a question, and submitting the answer. Figure 3.3 illustrates the basic information flow between the backend, the frontend and CP's controller.

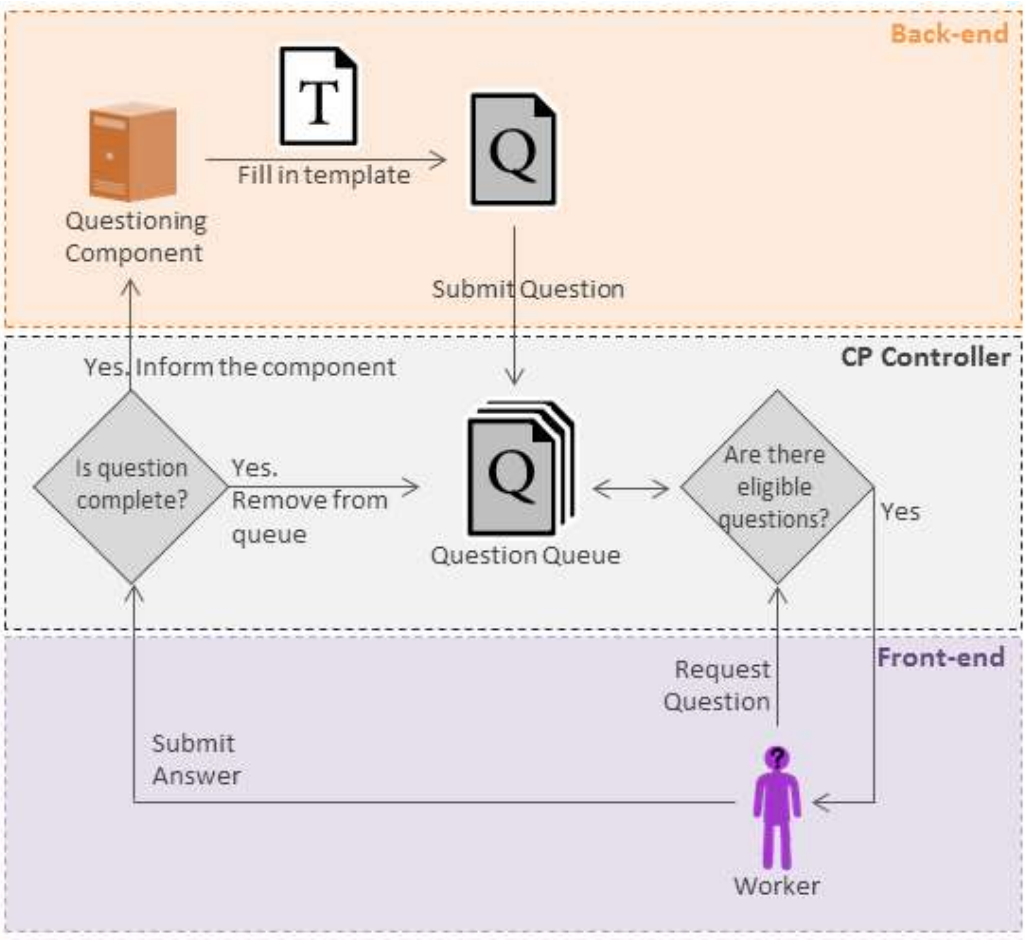


Figure 3.3. The basic information flow of the Crowdsourcing Platform.

The library offers a few Question Templates that developers can fill in with the question's text as well as the possible answers. CP is able to handle single-answer or multi-answer multiple choice questions, as well as multiple choice questions that can extend their options

through free-text input. Developers can also extend the library by creating templates for more question types. Each question is accompanied with some meta-information, such as who should be notified once the question is complete. Once a question has been formed it is submitted to CP Controller for handling.

The CP Controller maintains an internal queue of questions ready to be served. The web-interface sends requests for new questions on behalf of the worker. Figure 3.4 shows a screenshot of the web interface as it was implemented during the demonstration of ACIBa with an example question. When a question is requested, the CP Controller looks in the Question Queue for eligible questions, which in essence are questions that have not been asked to the same worker before. If one is found it is returned to the front-end that renders it in the browser. The answer (or the action of skipping the answer) goes back to the CP controller which collects it, and the cycle repeats with the next question. Each question requires the input of multiple workers before it can be considered as answered. As it was discussed in the literature review, that is typically a fixed number of answers for each question, but there are various other methodologies that can customize that number. Chapter 5 discusses this matter in more detail, and proposes a novel methodology that can determine an appropriate number of answers separately for each question.

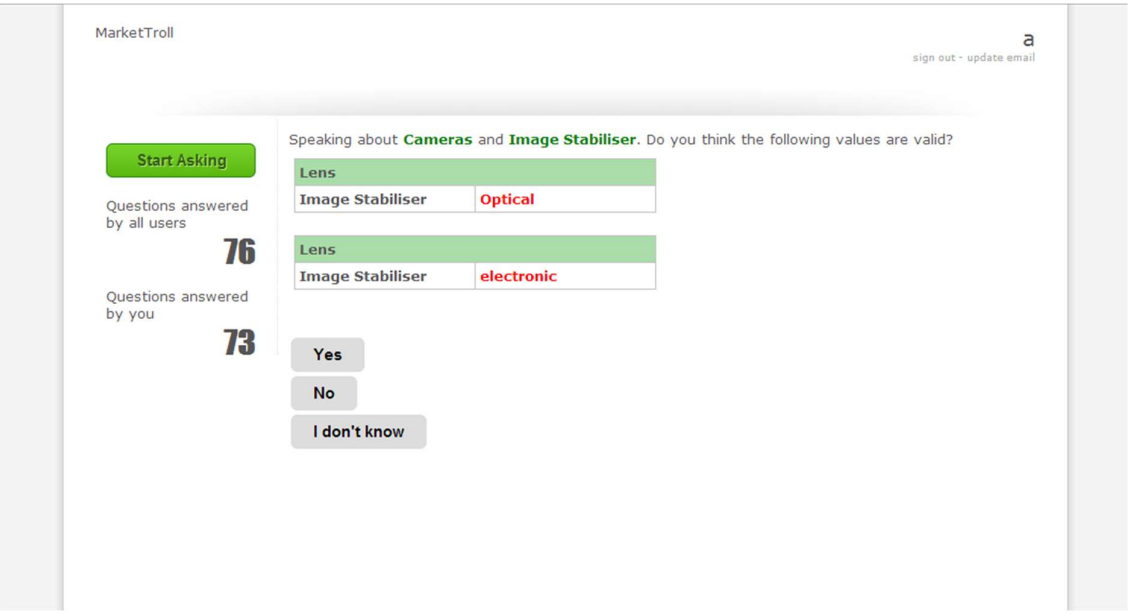


Figure 3.4. Screenshot of the Crowdsourcing System used during the implementation of ACIBa.

When a question gathers enough answers to be regarded as completed, the CP Controller will inform the "interested" component through a callback that new information has become available. This is a powerful mechanism, as it allows the Questioning Component to submit

the question and continue its execution without having to wait for the answer (which may take days to complete).

3.4 Product Analyser

The Product Analyser (PA) is responsible for transforming the data acquired from the Web Crawler to usable information. In an abstract description, PA is a two-step process. First, the individual product listings are analysed to form a reconstruction of the whole product category. The reconstructed category contains information such as which are the important attributes for this product, what values can they possibly take, in what units are they measured by and so on. When this information becomes available, the next step is to find the best values for each attribute for each possible use of the product. This is also the final outcome of PA, which will in turn be seeded to the Recommendation Engine.

PA implements the logic of the analysis through the Logic Controller and as such controls the flow of information. It performs that analysis though a well-defined, step by step procedure, discussed in greater detail in section 3.4.4. PA will attempt to acquire the required information with either hard coded, heuristic or artificial intelligence. Where this is not possible, PA will turn to the intelligence of the crowd. The Logic Controller (LC) oversees this process, and when needed generates questions in natural language, based on predefined question templates. CP then serves those questions to the crowd. When enough answers have been gathered, the question is deemed complete and a callback event notifies LC that this new information is now available so the execution can continue.

The next sections will present first the Bullet Classifier, the Attribute Merger and the Value Merger. All three components use artificial intelligence to achieve a portion of the analysis. Presented last is LC, so as to bind everything together in a meaningful flow of events.

3.4.1 The Bullet Classifier

Not every feature of a product bears the same weight when making a choice. For example, a typical Compact Digital Camera is small enough for the average consumer, so its weight and dimensions tend to not affect the decision as much as other features. One of the objectives of the Product Analyser is to identify the attributes important to the decision making process. To automatically acquire this knowledge, ACIBa utilises the short bullet-point lists that precede the product's extended description and highlight its most salient attributes. Figure 3.5 contains a snippet of a typical webpage that can help visualize this information. By examining the bullet-point lists of many products we can observe which attributes are encountered more often, and thus deduce which ones are important for the whole product category. Since bullets are written in natural language, the challenge with this task is to

develop an appropriate method that can correctly identify the attribute that the bullet-point text refers to.

Before we discuss the details of such a method, it is mandatory to understand the nature of bullets. Bullets mainly consist of either the value of an attribute, the name, or both. For example, when browsing for cameras, "4x optical zoom" explicitly mentions both the attribute's name (optical zoom) and its value (4x). However, in the bullet "24.4 megapixels" only the value is mentioned, while the attribute's name (sensor resolution) is implied. On the other hand "Waterproof" contains only the attribute's name and the value (yes or no) is completely omitted. Bullets may also contain connective or promotional words (such as: with, up to, new, amazing, etc.). These words constitute noise since they are not part of the training data. Some bullets may consist purely of noise words (e.g. Modern look, Amazing offer). Since there is no correct attribute for these bullets their classification is ipso facto impossible, introducing a baseline error rate.

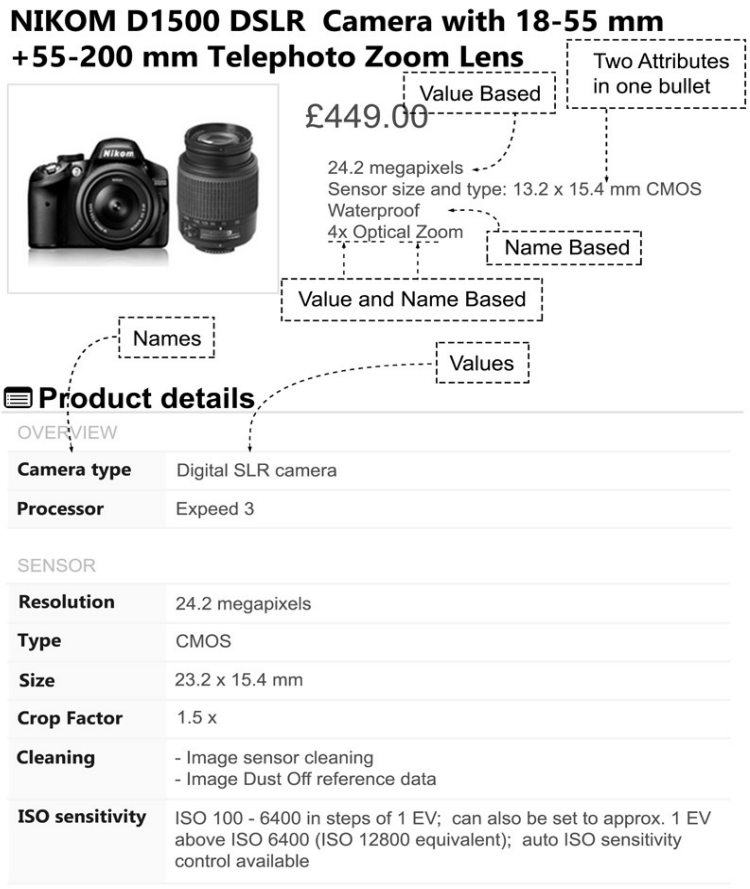


Figure 3.5. Snapshot from an actual webpage. Important areas are highlighted with dashed lines.

In essence, the task at hand is to classify each bullet to the attribute they refer to. The difficulty of the task lies in the fact that there are no training data readily available for such bullet-point texts, or else, there are no bullets for which there is already knowledge of their correct class (class and attribute have a one-to-one correlation in this context, each product attribute is also a class). With no known pairs of "bullet -> attribute", supervised classification methods cannot be used. However, what is available is the structured table of attributes that exists in the same webpages (Figure 3.5). In this table, parts of the bullet should exist either as an attribute's name, a value or both. For example, each part of the "4x optical zoom" bullet is potentially part of an attribute's name or value. As a result from the above, it can be concluded that the knowledge of where a bullet should be classified does exist in the table, but not in a convenient format to be used by typical supervised classification techniques. It seems that there is a gap in the literature on how to tackle the aforementioned problem and utilise training data that exist in a different format from the texts that need to be classified. At the same time using this data is crucial, as it allows the completion of the task without having to hand-tag training instances.

This section briefly introduces an ensemble classifier that can tackle this issue by using multiple different member classifiers. Each member classifier of the ensemble has a unique feature set and can be trained with different training data (i.e. only the attribute's names or values). This segmentation causes each classifier to be able to capture only a single "type" of bullet (i.e. name-based or value-based) but collectively they are able to identify any type. In essence, since there are no training instances for bullets as a whole, training instances that might contain just pieces of it are used instead. Due to its length, the detailed description of the ensemble classification methodology is discussed in chapter 4.

In general, a classifier will be given a value and it will calculate the probability of this value to belong to any of the possible classes. The probability function is dependent solely on the underlying classification model of each classifier (e.g. Naïve Bayesian Network). Specifically in this context, each individual classifier is given a bullet, and produces a score for every possible attribute, showing the likelihood of this bullet to refer to the respective attribute. The scores of the member classifiers are combined using a fixed weighted scheme creating the ensemble classifier. The highest scoring attribute is where the ensemble "believes" that the given bullet belongs.

To demonstrate this approach a prototype of the ensemble classifier has been built, namely the Bullet Classifier. It is based on four classifiers with unique feature-sets specifically tailored for consumer electronics. They have widely different feature sets, are implemented by different classification models and are trained with different parts of the data. A brief

introduction of each classifier is as follows, while detailed descriptions are presented in Chapter 4.

- The Name Classifier (NC) is able to find similarities between the bullets and the name of an attribute. In essence it is examining if the bullet contains any words that also appear in the name of the attribute. The reason NC is not sufficient alone is that some bullets will not contain any words that refer to the name of any attribute (e.g. 24.4. megapixels). In such case NC will not give any scores.
- The Value Classifier (VC) is capturing the similarities between the bullet and the typical values of an attribute. It examines over a hundred parameters such as the occurrence of lowercase Latin letters, numbers contained within certain ranges and some meta-data such as the total length of the bullet and the total number of upper-case characters. The exact opposite of NC, VC is not sufficient alone because some bullets are constructed without any reference to the values of any attribute (e.g. waterproof). However, due to its generic feature set VC will always produce a score.
- The Units Classifier (UC) is attempting to find the units in which an attribute may be measured, such as "cm" or "megapixels". In general only one or a handful of attributes will be using the same units, and that makes UC very effective in identifying value-based bullets that contain units. However, not all attributes use units, and as a result UC does not always produce a score.
- The Special Words Classifier (SWC) counts the occurrences of letters and numbers where special use is assumed, such as in model numbers or versions (e.g. “DVB-T2”, in “DVB-T2 Freeview HD Tuner”). As these are often absent, SWC does not always produce a score.

Looking back at the greater picture, the Bullet Classifier will attempt to match each bullet to an attribute. Each bullet is passed through each of the above classifiers, who by examining different criteria, produce a score for each attribute. The scores are added together using different weights for each classifier and the final attribute selected is the one with the highest combined score. This is performed for every bullet gathered by the Web Crawler, producing an equal number of attributes where bullets are perceived to refer to. The attributes are counted and ordered according to their occurrence, which is the final product of the Bullet Classifier. This information is used as a metric of the attribute’s importance in the decision making process.

3.4.2 Attribute Merger

Webpages, even from the same website, do not always have a uniform structure for the product attributes. As a result, the same attribute might be found with a slightly different name, or grouped under a different category. For example, the LED light of kettles can be found as "LED light", "LED Indicator" or just "LED". The category where this attribute is placed is not always the same either, for example some kettles will list the LED light under "extra features", some under just "features" and some under "other". Any combination of the above category and name will create a unique class, and eventually there is the need to consolidate those classes back to one. Merging similar attributes is not as simple as looking at the attribute's name alone. The category (hereinafter presented in brackets) plays a crucial role in separating attributes. For example "(lens) size" and "(body) size" genuinely refer to different things while they share a name and differ in category, while "(features) LED light" and "(other) LED light", don't.

The Attribute Merger will attempt to identify and merge pairs of attributes. This is achieved by using the classifiers previously mentioned to create a metric for the similarity between attributes. This is a multistep process that begins with the raw classification results, and proceeds with tuning and handling of special cases.

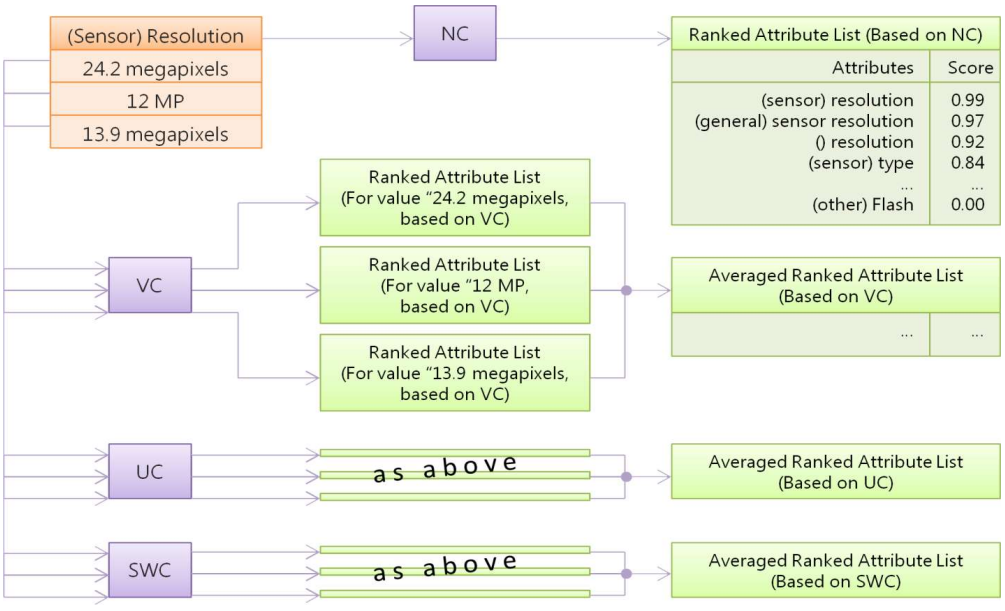


Figure 3.6. Classification of attributes during merging with other attributes. The orange box represents a single attribute with a few possible values. The purple boxes represent the member classifiers. The green boxes represent a list of all the attributes (which naturally includes the attribute from the orange box). Each attribute in the green list is accompanied by a score, showing to what extent the classifier thinks that the classified text belongs to that attribute. The top right box shows an extended example of such list. To simplify the diagram, where there is repetition green boxes might contain less text; however all green boxes have the same internal format and essentially it is only the scores that change.

1. Initially, the name and category of each attribute go into the Name Classifier (NC). NC is able to detect similarities between the names of attributes, if these contain the same words. The result of the classifier is a ranked list of all other attributes, ordered according to the perceived similarity with the name of the examined attribute. Each value of the attribute goes through the Values Classifier (VC). VC is able to examine a given value and discover other attributes with similarly looking values. The result is again a ranked list of all other attributes, ordered according to the similarity score as this is perceived by VC, for the given value. As each value of the attribute produces a different list, these are combined into a single list that contains the average from all values. The values go through the same process for the other two classifiers, one that detects similarities in units (UC), and one that looks for any special words that might have been used (SWC) such as acronyms or models. The final outcome of these steps is the four lists on the right of Figure 3.6, each containing the similarity score as perceived by the respective classifier. Of course this process is repeated for each attribute, producing 4 lists per attribute, which collectively show the similarity of every attribute to every other attribute.
2. Before the raw result tables can be used some special cases need to be handled. An initial issue to anticipate is that unrelated attributes with boolean values will look exceptionally similar. For example "(accessories) charger = yes", and "(general) built-in flash = yes", both typically have "yes" as a value. VC will produce exceptionally high similarity scores for any pair of boolean attributes, overshadowing the result. To handle those cases, the similarity score for pairs of boolean attributes is capped to a logical estimate. Pairs of non-boolean to boolean attributes are obviously mistaken, so their similarity score is also capped. The reason these cases might occur in the first place, is that because the words "yes" and "no" are very generic, they might easily match other values that are not boolean. Attributes under the same category will also get artificially increased similarity scores. For example, as far as NC is concerned, "(sensor) type" and "(sensor) size" are 50% the same, while in truth they refer to a different thing. For that reason, pairs of attributes that belong to the same category have their similarity score recalculated using only the name of the attribute (for the last example, "type" and "size"). Finally, attributes whose names contain ‘min’ or ‘max’, typically have all other parts of the attribute completely the same, including the name, the category, the units they are measured by as well as typical values. As these

attributes should obviously not be merged, any opposite pairs found have their similarity score reset to 0 for all 4 classifiers.

3. Once these special cases have been handled, the four lists can be merged to one. Because each classifier produces numbers in a different magnitude, weights are used to combine the results of each classifier. The weights are calculated empirically, while more details on this can be found in section 4.2.4. With the results combined, each attribute now has a single list that connects it to every other attribute. If put together, these lists create a matrix showing the similarity of every attribute, to every other attribute. It is worth mentioning that the matrix is not symmetric, as the similarity of attribute A to B, is not necessarily the same as B to A (though they are very likely to be close).
4. The next step is to tune the results in the matrix. Occasionally there are classes that will match more attributes than normal, or attributes that are being matched by more classes than normal. To identify those cases the average of the whole matrix is calculated. Then any row that has an average above this global, is tuned down. This happens by multiplying the similarity scores of the row by `Global_Average/Row_Average`. This brings those classes back in order, while maintaining the differences between the scores of the row and thus preserving the important information. If the row had more or less similar numbers and all were just elevated, now they are all back around the average. If a few of those numbers were really high, these will still be above the threshold to be considered for merging. The exact same procedure is repeated for columns.

The final matrix is checked for values above a threshold. Attributes in the diagonal are ignored as they refer to the similarity of an attribute with itself. If a pair exists in one direction the other direction is ignored. The identified pairs are sent to the crowd to validate whether attributes should indeed be merged or not, effectively reducing the amount of pairs that should be examined.

This approach is expandable for any number and type of classifiers, whether they are part of the bullet classifier or not. Any new classifier that can identify useful similarity information can be added to this approach. If the classifier operates on the names then it is added in a similar manner to NC, while if it operates on the values then it is added with the intermediate averaging step like VC.

The issue tackled here is in essence the same of consolidating database fields from different databases. The classifiers look at the title of the columns (the names), they look at the contents of the columns (the values) from a few different angles, and then produce a

similarity score for each pair of columns. Of course the classifiers used here have been developed specifically for consumer electronics, but it can be reasonably expected that with the appropriate classifiers this approach could be applied to a different context.

3.4.3 The Value Merger

Similar to the issue of multiple attributes existing many times with different names, values of the same attribute might also suffer from the same problem. For example is "USB 3.0" the same as "USB3"? Values like those belong to the same attribute and even though slightly different in truth they refer to the same thing. This becomes more common after the Attribute Merger has brought together values from different “styles” of attributes from different sources. The Value Merger aims to identify pairs of values that look similar. It operates in a very narrow space; just the values of a single attribute, and only for non-numerical attributes, which is attributes whose values are described by text (e.g. LED indicator) and not measured by a number (e.g. 400W).

To achieve its goals the Value Merger will make use of the member classifiers of the ensemble used at the Bullet Classifier. Because the problem at hand is different the ensemble cannot be used as a whole. To start with, the Unit Classifier (UC) cannot be of any help, since the values are textual and no units are expected (values of the same attribute would have the same unit anyway). The Value Classifier (VC) is not of great help either as it is very generic. It is designed to identify values that belong to the same attribute in the first place, and not discriminate between them.

The Name Classifier (NC) can be used to identify values that contain the same words. The design of NC allows it to adjust its feature set based on the given training set. As it will be described in greater detail in section 4.2.3.1, the feature set of NC consists of the stems of all the words found in the training set. For the needs of the Value Merger, the training set is the values themselves, and so the feature set will become the stems of the words found collectively between all the values of the same attribute. This allows NC to identify the same words in different values. The Special Words Classifier (SWC) can also be used to identify any special words (e.g. acronyms, version numbers, etc.) that are used by more than one values.

Both classifiers then classify all values, with the possible classes being all other values. Much like the Attribute Merger, this creates 2 matrices showing the likelihood of each value to be the same with any other value. The matrices are added to a single matrix from where pairs of values can be picked. Very high scoring pairs can be merged automatically, while high scoring pairs can be sent to the crowd for validation and low scoring pairs can be ignored.

3.4.4 Logic Controller

The Logic Controller (LC) is the module that guides analysis and dictates the information flow. LC is structured as a series of steps that can lead from the initial data acquired by the web-crawler, to the information the recommendation engine needs. The performed analysis is based on a combination of Artificial and Human Intelligence, with the latter being offered by the Crowdsourcing Platform. In essence this is the heart of the whole Product Analyser, as it dictates what information is required to perform the analysis. The remaining of this chapter will examine step by step all the gathered information.

In brief, the analysis starts with workers identifying possible usage scenarios for the product. They are also validating the importance of the attributes identified by the Bullet Classifier. The attributes that are acknowledged as important undergo further analysis, trying to determine the possible values they can take. If they are numeric, the focus is on the range of values they can take and the units they are measured by. If they are textual, the focus is in finding the possible options they can take (e.g. if the attribute is "(sensor) type", identify all the possible sensor types). Once this information is known, the crowd is asked to pick the best values for each use. The best values are the outcome of this module, and can then be used for scoring products, producing recommendations.

In detail, the procedure goes through the following steps.

1) Identify Usage scenarios

A single question is used asking the crowd to identify possible usage scenarios of the product. Workers get the option to vote on answers of other workers, or fill in their own suggestions. This allows the question to be used for any product category, since it is the crowd that is determining the possible usage scenarios. The option "Typical everyday use" is automatically seeded as an option, giving workers at least one option they can vote for, without limiting the application to a specific category. Under this voting scheme the acceptance of the proposed uses is not very straightforward; some use scenarios might have been suggested early on, and thus had more chances to accumulate votes. A workaround around this issue would be to accept uses based on the proportion of votes they got after they got proposed, but a more suitable solution would be to break this question in two, one that identifies possible uses, and one that validates those uses through the votes of the crowd.

The possible uses are crucial to the end result, since they are the only customization options presented to the final user. ACIBa will aim to produce different recommendations per use scenario, assuming that the crowd will identify that each use scenario has different needs,

and that different products are better suited for those needs. This question can run alongside questions 2-5, as the outcome information will not be required again before step 6.

2) Determine the importance of attributes

The original data gathered by the web crawler are combined in a list of attributes and their possible values. The ensemble classification procedure previously described is used to trim down the full list to a more manageable number, by selecting those that are most often featured in bullet points. These can then be merged with the most commonly found attributes, as these occur by directly counting their repetition between webpages (in the tables of attributes). This acts as a safety net mechanism, in case the previous methodology failed to properly identify the correct attributes.

Workers are presented with the pre-screened list of important attributes, and they are asked to rate the importance of each one between 1 and 5, which corresponds to scores 0 to 4. The scores are then averaged and brought to per cent. Attributes with a voted importance over 60% are deemed important. This threshold is a trade-off between not taking into account potentially important attributes, and taking into account potentially unimportant attributes. With 60% leaning towards the later, safer end.

This step can be used to validate the results of the Bullet Classifier, whose performance can be measured by the portion of attributes that the crowd has actually validated. This can also be used to evaluate the approach of using bullet points to identify important attributes, by comparing how many of the crowd validated attributes originate from the Bullet Classifier and how many from the list of most common attributes. Of course nothing guarantees that either list contains the actually most important attributes, but the results can still serve as a measure of comparison.

3) Merge similar attributes.

The original list of attributes might contain the same attribute multiple times, if that has appeared with a different name or under a different category. Such attributes need to be consolidated. The Attribute Merger described in section 3.4.2 is used to identify possible pairs of attribute that might suffer from that issue. Those pairs are then sent to the crowd for validation, in a straightforward question that asks if attribute A is the same as attribute B. If the crowd validates the relation then the attributes are merged. This means that their values are grouped into a single list, and a note is made of possible names this attribute might have. More than two attributes can be merged together, for example if A is merged to B and B is merged to C.

Merging attributes that have not been marked as important would be a waste of resources, as these attributes are effectively unused for the rest of the analysis. However,

merging an important attribute with an unimportant attribute can still be useful, as it can potentially provide a wealth of information in the form of additional attribute values. As a result of the above, merging pairs are considered only if any two attributes have a similarity score above a certain threshold, and either of them is in the list of identified important attributes. For this reason, the questions of this step can begin as results from step 2 become available.

4) Define the type of attribute

The result of the previous step is a clean list of important attributes. This step is looking to categorise the attributes in two types; numerical and options-list. This is achieved through a question asking workers to categorize the attribute as either one or the other. Numerical attributes should be a number, possibly followed by units. Options-list attributes are those, whose values can be one or more options from a predefined list of options (e.g. flash-type can be pop-up, built-in or hot-shoe). Not all attributes fall under these two categories, and this is the reason the separation is not happening automatically. However, within the scope of this research LC is only capable of handling these two types, enough to create a prototype system and cover the most common cases. The generated question also offers the option of "other", if the attribute does not fall under either of those categories. In that case the attribute would be disregarded, but a fully operational system should consider representing other types of attributes as well, such as a number range or mutually exclusive options. Step 4 can begin when step 3 can no longer make merges for this attribute.

5a) Determine the values of Numerical attributes

Numerical attributes are analysed automatically to determine the range of possible values they can take, and the unit they are measured by. Different expressions of the same units are merged automatically based on a dictionary, e.g. "inch", "inches", and the symbol for inches ("). The most commonly occurring unit is then considered the main unit of the attribute. Units who's relation can be identified are converted to the main unit of the attribute (e.g. if "mm" have been identified as the main unit of the attribute, values in "cm" are converted to "mm" as well). Remaining values in the non-main unit are discarded as they cannot be used.

In order to allow workers to vote on the best number (at a later stage), the individual numbers gathered need to be converted to a continuous space. This can be achieved through a process called discretization, which in essence transforms a continuous attribute to categorical. There are various discretization methods applicable, but the issue at hand has the requirement to create discrete options that preserve the initial space of the numbers found. These options should be able to properly reflect the values and thus cannot be hardcoded since different attributes are expressed in completely different ranges. For example the

weight of a TV is roughly between 3 and 10 kg, but kettles need about 2000 to 4000 W. An additional requirement is to preserve more detail around values that are found more often, and group more rare values together. Equal frequency binning can be used to honour both these requirements.

Equal frequency binning splits the available numbers in a predetermined number of bins. The numbers are ordered and each bin gets roughly the same amount of numbers (thus its name). At the end, each bin can be characterized by the smallest and the biggest number it contains. To avoid having a bin filled with a single number, only the unique numbers identified are used during discretization. This also prevents a single number from dominating the results. To give a quick example of the equal frequency binning, assume the following number set of 12 unique numbers, [4,5,6,7,8,9,10,15,20,25,30,40]. "Discretizing" the set with equal frequency binning, in 3 bins, would require each bin to have 4 numbers. This would produce the following result: bin1=[4,7], bin2=[8,15], bin3=[20,40]. In this example, bin1 spans 4 units, while bin2 spans 8 and bin3 spans 21. This allows more detail where there is higher concentration of numbers, as the first 12 units (where is there higher concentration) occupy 2 bins, in comparison the later 20 that occupy the remaining 1.

In order to cover the whole spectrum of numbers the bins are expanded to account for the numbers in between bins, and numbers smaller and bigger than those originally identified. Without loss of generality we expand the bins to the right, and add two extra bins to cover both ends of the spectrum. The bins of the previous example would become (<4)[4,8)[8,20)[20,40](>40). This allows to capture numbers that might have not been initially included in the data mined numbers. If a worker votes that the best range is [8,20), then this includes any number in that range, even though not every number has been originally discovered. In essence, the bins are a reconstruction of the possible values the attribute can take.

5b) Determine the values of Options-list attributes

Options-list type attributes follow a different path. Initially values are examined for patterns (such as "ISO 100 / ISO 200 / ISO 400 / ISO 800"). If a pattern is found, the attribute is automatically split at the key symbol (in this case the slash) and each part is treated as an individual value. To identify a pattern, the resulting number of characters in each part is examined. If the character is able to split the value in roughly equal size phrases, then this qualifies as a pattern. In the above example when split at the slash, each part is exactly 7 characters long creating a very strong pattern. However, other texts might not be as obvious, For example "carrying case, strap, 2xAA batteries" can be logically split at the comas but the result varies significantly in length. Once again the intellect of the crowd is used to solve this

issue. Values that do not meet the criteria to be split automatically are examined for possible split points and sent to the crowd for validation.

The final list of (now split) values are checked for similarities so they can be merged. Of course values that are exactly the same are merged automatically, but similar looking values are being sent to the crowd for validation. In addition, when the crowd is later given the option to vote for the best values (described below), they can propose pairs of similar values for merging that the system has not automatically marked as possibly similar. If a pair is pointed out by a worker a validation question is generated, identical to those created automatically by the system. The 'best values' question (described below) is halted until a decision is made.

6) Find the best values

Once the values for both types of attributes have been determined, the crowd is asked which value is best for each use scenario. The question needs to be phrased in a way that does not confuse workers into voting for the absolute best value, but for the value that is best for the intended use of the product. This is important because the angle of the recommender is to replace the expert, and it can be reasonably assumed that a good advisor would not recommend the top-of-the-line product as a blanket recommendation for everyone. To illustrate that with an example, the best CPU produced might be the absolutely 'best', but for typical office use that CPU would be heavily underutilised so it is probably not the best choice. Workers are allowed to vote for multiple values, as often more than one values are acceptable. This is the last step of the Logic Controller and the results can be passed as-is to the Recommendation Engine, which will then use them to score products and produce recommendations.

It is worth noting that questions 1, 2 and 6 do not have objectively correct answers, and seek to extract the crowd's knowledge, not intelligence. The distinction is clear enough that opens up the possibility to be passed through a different crowd. Marketplaces for such tasks often allow to set worker eligibility criteria, in which case the questions can be targeted to only those workers with the relative knowledge.

3.5 Recommendation Engine

The Recommendation Engine (RE) is the final module that binds all the information together to produce the actual product recommendations. RE takes as input the database with the products that need to be evaluated. The database has to contain at least the important attributes that have been previously identified, as well as ratings, number of reviews and prices. The second input comes from the Product Analyser (PA) which produces 3 distinct results. The possible usage scenarios, for which scoring will be done separately, the

list of important attributes which are the only ones that will be scored, and the best values for each of those attributes, separately for each usage scenario. With these information, RE scores all the products in the database. The score of the product then becomes another attribute, along with price, rating and number of reviews. The final input to RE are groups of filters that highlight different attributes of the products, such as “lowest price to score ratio”. The filtering is applied to the scored results and recommendations are produced.

Scoring products is different for numeric and option-list attributes.

For option-list products, the result from PA is a list of the possible attribute’s values (the options) and their scores, as identified by the crowd. For example the options of the attribute "extras" for portable speakers might be "Carrying strap: 28, Carrying case: 13, Hanging hook: 3". Each product that needs to be scored acquires the points of each of the options it has from the possible pool of options. To continue the above example, if a particular speaker's extras are "Carrying strap and Carrying case", it will get a score of 28+13=41.

Numerical attribute scoring is more complex. The result of PA is bands of values and their respective scores. The bands are created automatically from the LC using equal frequency binning and as a result have different widths (3.4.4). Figure 3.7 shows an example of such results in a bar chart. In this example, cameras with 11.9MP belong in the band of [10-12)MP, and as a result should be given a score of 45. At the same time, cameras with 12.1MP belong in the band of [12-15)MP, and should be given a score of 30. Scoring numeric attributes using the bands, causes a really small change in the value to create major differences in the score. As this is an unwelcome quality, the histogram line is used instead to smooth the differences between bands to create a fairer scoring methodology. Because the first and last bands are expressed as "less than" and "more than", their range is not specified. For these two bands, the histogram extends half the width of the attached band, following the naturally created line. To continue on the previous example, the histogram line that runs from (11,45) to (13.5,30) is given by the function `f(x) = -6x + 111`. Using that function, 11.9MP is awarded a score of 39.6, and 12.1 a score of 38.4, fixing the previous problem.

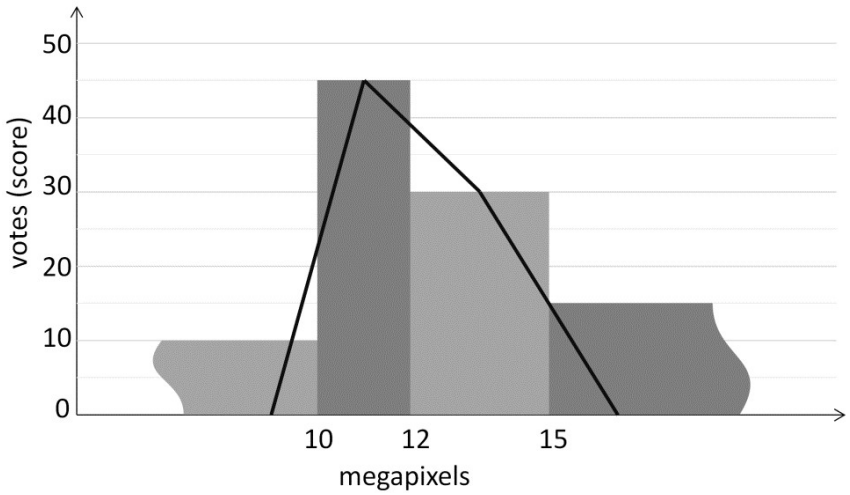


Figure 3.7. Example scoring histogram for the attribute "(sensor) resolution". The edge bars have no limits, as they represent "less than 10" and "more than 15". For that reason the histogram is drawn using half the width, of the adjacent bars.

Before the retrieved score can be used, it needs to be adjusted to account for the following phenomenon. Assume CameraA costs £100 and offers 19MP, and CameraB costs £200 offers only 11MP. According to the histogram of Figure 3.7, CameraA will receive a score of 0 and CameraB 45, even though CameraA offers a higher resolution at a better price. This phenomenon is created because workers are asked to vote which value is good enough for a given use, and not the absolute best (which after all could be inferred by other means). The rationale behind this, is that if there is no reason for a camera to have 25MP, then it does not worth paying for it. That been said, if a higher value product can be acquired at a better price then it should be promoted. To adjust for this phenomenon, scores are boosted in the following way.

Each product's price is tested against the prices of all lower bands, to find the percentage of more expensive products in each band. Then, the product receives a bonus to its score, equal to the score of all lower bands, multiplied by their respective percentage. Practically, this means that if a product is cheaper than every other product in a lower band, it will receive that band's score as well. If a product is more expensive than every other product in the inferior band then no extra score is added. To complete the previous example, assume that CameraA is cheaper than 30% of cameras between [10-12) MP. It would then receive an extra score of 30% × 45 = 13.50. This process is repeated for the other 2 lower bands (i.e. (<10) and [12-15)).

Each product needs to be represented by a single score. However, the occurring scores for different attributes might be expressed in different magnitudes, especially since option-list

and numerical attributes originate from two very different scoring techniques. In order to become comparable they need to be transferred to the same space. This is achieved in a 2 step procedure.

Table 3.2 shows an example transformation of 3 products with 4 attributes. Table cells represent scores of attributes and have been exacerbated to highlight the reasoning of these steps. Logically, Product 3 should clearly get the highest score, since it has a huge advantage in Attribute 4 that has the highest weight, while the differences in Attributes 2 and 3 are so small that they should be insignificant. However, because the scores for Attribute1 are expressed in a different magnitude it dominates the result. This is not because products had a greater difference in Attribute 1 neither because Attribute 1 is more important, but merely because of the sheer score numbers Attribute 1 has.

1. To solve this issue, attribute scores are transferred from their existing range to a range between the minimum value and 100. Of course this transformation cannot be applied directly to the raw values. This would create huge differences in attributes that have small ranges, changing the actual information they carry. At the same time, attributes whose values are over 100 would become reversed. To counter this issue, all values are transformed so that the average of the attribute across all products equals 100. This is done by multiplying each value by `100/attribute_average`. This way, the relative difference between products, which is the key information, is preserved.

2. The transformed numbers are then shifted to a space between the attribute's lowest score and 100. The shift is made with the affine function `f(x)=ax+b`, where `a=(100-min)/(max-min)` and `b=min-(a * min)`, where min and max refer to the smaller and highest score of the attribute across all products.

Table 3.2 shows this transformation step by step. As it can be seen attributes 2 and 3 have minor differences between products so they should not greatly affect the final decision. This is one of the qualities that the transformation function needs to retain. Indeed in at the end of step 3, score differences of Attributes 2 and 3 remain insignificant as they ought to. Attributes 1 and 4 have retained their relative differences, but the numbers are now comparable and can be used for scoring. The weights of the attributes originate from the importance of attributes given by workers at the 2nd step of LC (3.4.4, p.36).

Table 3.2. Example transformation of score values, for 3 products with 4 attributes.

Initial scores for each attribute.

	Weight	Product 1	Product 2	Product 3
Attribute 1	0.10	1000.00	850.00	640.00
Attribute 2	0.20	1001.00	1000.00	999.00
Attribute 3	0.30	1.01	1.00	0.99
Attribute 4	0.40	2	15	44
Final score (Sum of Weight × Value)		300.90	291.30	281.70

Transformation of values so the average of each attribute is 100 (step 1)

Attribute 1	0.10	120.48	102.41	77.11
Attribute 2	0.20	100.10	100.00	99.90
Attribute 3	0.30	101.00	100.00	99.00
Attribute 4	0.40	5.00	75.00	220.00
Final Score		64.37	90.24	145.39

Transferring values to the [min-100] space (step 2)

Attribute 1	0.10	100.00	90.46	77.11
Attribute 2	0.20	100.00	99.95	99.90
Attribute 3	0.30	100.00	99.50	99.00
Attribute 4	0.40	5.00	35.93	100.00
Final Score		62.00	73.26	97.39

Once the transformation of scores is completed and scores are comparable to each other, the weighted sum is calculated and used as the score of the product. This score serves as an index of fitness for a product for a particular use, and not as a metric for the absolute best product. The remaining work of the Recommendation Engine is to apply filters, which can be used to select the final products to recommend. Filters can be mixed together and prioritized in order to create good and diversified recommendations. The main filters available are the score, rating, popularity (i.e. number of reviews) and price. For example, some recommendations could be the "Highest absolute score with at least 4 stars rating and 10 reviews", or the "Best score to price ratio, with at least 4 stars and 10 reviews, from the top 1% of products (according to their score)".

3.6 Summary and Conclusions

This chapter has discussed the various details of the proposed recommendation framework. As we have seen there are many different parts that need to work together towards the final result. This research does not emphasise equally on each aspect. The main objective to be met was the development of a proof-of-concept system that can produce recommendations, so the full cycle could be demonstrated.

The core of the information extraction happens though AI coupled with crowdsourcing. In essence, AI tries to do the bulk of the work while human intelligence is used for the fine detail. This "hybrid" approach takes the best of both worlds, as on the one hand it allows to process large quantities of information, and the other it has the potential to produce very accurate results. This work is tuned for consumer electronics, but the same principles can be applied for any product category, where the decision can be based on some objective truth over subjective preference.

A big portion of the work is devoted in the development of the ensemble classification technique. The novelty of the approach is its ability to work in the absence of traditional training data. This chapter briefly presented the member classifiers of the ensemble created to classify bullet points, within the greater context of identifying the salient attributes of a product category. However, the same principles can be applied to develop a similar ensemble for completely different scenarios that face a similar situation. The details of the developed member classifiers, along with an evaluation of the technique are presented in Chapter 4. The member classifiers have also been used in other integral components, i.e. the Attribute Merger to identify similarities between attributes, and the Value Merger for similarities between an attribute's values.

The Logic Controller (LC) has also occupied a considerable part of the work. It dictates the intermediate information that needs to be acquired until the final knowledge extraction questions can be asked. In essence LC guides the analysis. An important quality of LC is that the intermediate steps, even though chained, can be altered independently. For example, to determine which were the most important attributes for decision making we used the attributes highlighted by the markets and the attributes that were almost always found in product specifications. However, other ideas could also be tested, such as which attributes are offered as filters by the markets, or which attributes form their own product subcategories. Should one wish to test those ideas, step 2 of LC can be changed without affecting other parts of the components, and important design quality.

LC achieves most of its goals through the Crowdsourcing Platform that interfaces between the rest of the software and the crowd. The platform offers an API to create

questions, and event listeners that allow actions to happen when questions finish. This creates a dynamic environment where the generated questions are often based upon the answers of the previous questions. An important issue that naturally arises is how many answers need to be collected before a question can be regarded as 'finished'. Presented at chapter 5 is a novel approach that can significantly reduce the number of answers required, without compromising accuracy (in comparison to the fixed-number of answers approach).

For the scope of this work data acquisition is performed through bespoke information extractors tailored to specific websites. This approach can produce very accurate information but requires effort to scale. The greatest limitation is the creation of a normalized database of prices, reviews, and detailed product attributes. The main challenge is not the extraction of the data per se, but rather the identification and consolidation of information from different sources. Use of commercial APIs allows this work to be at the very least feasible, though there are ways that allow for the manual creation of such database.

The recommendation algorithms focus on how to score different attributes so their scores become comparable. The recommendations themselves can then occur by simple mixing of criteria at will, e.g. highest score at the lowest price. This work does not seek to answer which criteria create the best recommendations, as this is subject to user preferences and might require extensive market research to discover. However, it has become clear from the relevant literature that recommendations for the same thing should strive for diversification, with each recommendation offering something different.

This framework allows to transform the raw data found in product web pages into usable data. It taps into the collective intelligence of the crowd to analyse those data, as well as acquire information about the product category. The end product is a small list of recommended items, which are all relevant to the user's needs, while at the same time can be diverse enough to constitute meaningful alternatives. But that is not the only outcome; the system has also produced the knowledge of why these products are better. By allowing clear and transparent procedures that lead to a product being recommended, there is a great potential to improve user's trust. We believe that the combination of those factors can effectively assist in the problem of information overloading in online markets.

4 THE ENSEMBLE CLASSIFIER

Training data are not always readily available and often expensive to acquire, a cost that can potentially be avoided if the required information could be obtained from an existing data source. This work proposes the use of an ensemble of classifiers that can tap into different data sources for their training, but can collectively classify the required instances.

The first half of this chapter will present the methodology that creates the ensemble classifier used for the bullet-point texts of consumer electronics; previously called the 'Bullet Classifier'. It will discuss data filtering, the details of the member classifiers and finally the created ensemble. The second half (4.3) presents the evaluation of the Bullet Classifier, where it is determined if it is suitable to solve the issue of properly identifying the most commonly appearing attributes.

4.1 Introduction

A classifier is in essence a function that maps an unlabeled object to a label (or else a class). The supervised classification model is a way to create such function from a set of already labeled objects, a process called training. Different classification models have different internal structures and achieve their goal using different principles, but the operational use of the resulting classifier is virtually the same. A classifier is first trained with a set of labeled data; once trained, the classifier is presented with a new, previously unseen object, and is called to label it.

Each classifier examines a number of features of the object that needs to be classified in order to make a judgment. Features are in essence descriptors of the classes, chosen for their ability to discriminate between them. For example, if a classifier attempts to classify animals into birds, mammals and reptiles, a good descriptor would be the existence of feathers or the existence of fur, and a really bad descriptor would be the number of eyes, since it does not have any discriminating power. A classifier can be described by a vector of its features, $F_{[1\text{-}n]} = (f_1, f_2, \ldots, f_n)$, where n is the number of features. For example the feature vector of the above classifier would be $F_{EXAPMLE[1\text{-}2]}$(“has feathers”, “has fur”). Each object that needs to be classified has to be expressed in terms of the feature vector, producing a signature. For example the signature of a dog would be (false, true), since a dog does not have feathers (f1 is false) and has fur (f2 is true). Abstracting the inner-workings of a classifier, if the training set contains enough items with that signature that are mammals, the classifier will then learn that other objects with that signature are also likely to be mammals.

One of the main limitations of supervised classification is the requirement for training data, which might not always be available or be expensive to acquire. Chapter 3 (section 3.4.1) illustrated this issue through the example of bullet classification, where no training

data were available. Section 3.4.1 has also discussed how the essence of the information contained in training instances, i.e. to which attribute the bullet point refers, can be found in the detailed table of attributes. What is required, then, is a methodology that can utilize this information to achieve classification, without requiring the manual creation of training instances.

This chapter proposes a novel ensemble classification approach that is based on the creation of multiple different classifiers. Each classifier has a unique feature set that allows it to correctly classify only some of the unlabeled instances. If there is enough diversification between the classifiers, each one will be able to classify a different group of instances, and as a result they can collectively classify most of the instances. The main benefit of that approach is that such classifiers can be trained from other data sources and they don't require the labeled and unlabeled instances (i.e. the training data and the data that need to be classified) to be in the same format.

For this work training data are acquired from the structured table of attributes of product webpages. It should be noted that examples are drawn mainly from digital cameras for consistency, but the classifiers are not specifically designed for cameras. Each attribute translates to a triplet consisting of the category name, the attribute name and the value. The category is a heading in the table of attributes grouping many attributes that refer to the same thing. For example, a few attributes from Figure 3.5 (p.28) are the following:

- (sensor) resolution = 24.2 megapixels
- (general) size = 14x16x6cm

Each class is created by a pair of category and attribute name, e.g. "(sensor) size". The category is essential because the name alone can often be ambiguous, for example the attribute "size" can refer either to the sensor or to the frame of a camera. If no category is found then none is used. Each attribute (and thus each class) can have multiple values, as these originate from different product webpages. As a result, a more realistic example of the extracted attributes looks like this:

- (sensor) resolution = [24.2 megapixels, 12MP, 18.1MP, 20.1 Megapixels]
- (general) size = [14x16x6cm, 12x18x6cm, 4 x 5 x 2 inch]

The values to be classified are bullet point texts (bullets) also extracted from product webpages. Bullets are vaguely constructed using either the name of an attribute, the value, or both. To utilize the multiple pieces of information that might be contained in a bullet, four classifiers have been developed. They have widely different feature sets, are implemented by different classification models and are trained with different training sets. The combined

feature set of all classifiers is able to capture a wide range of characteristics of the bullets. In essence, a bullet might contain data that exist in any training set, and one of the member classifiers should be able to identify it.

4.2 Methodology

4.2.1 Introduction

The novelty of this ensemble classification approach lies in the use of multiple different classifiers, rather than diversified versions of the same core classifier. Each unlabeled instance goes through all classifiers in parallel. The outcome of each classifier is a list of all the possible classes, accompanied by a score which shows the likelihood of the instance to belong to that label. The scores are added together using predetermined weights for each classifier, and that produces a similar list showing the scores of the ensemble. Figure 4.1 illustrates this process with an abstract example. All classifiers should have the same outcome classes.

The ensemble classification approach has been tailored to meet the needs of bullet classification. Four classifiers have been developed targeting specifically the bullet point texts of consumer electronics products, and using different parts of the table of attributes as training data. Section 4.2.2 goes through the filtering steps of the acquiring data. Section 4.2.3 presents each classifier in great detail, while section 4.2.4 discusses the ensemble and the weight selection.

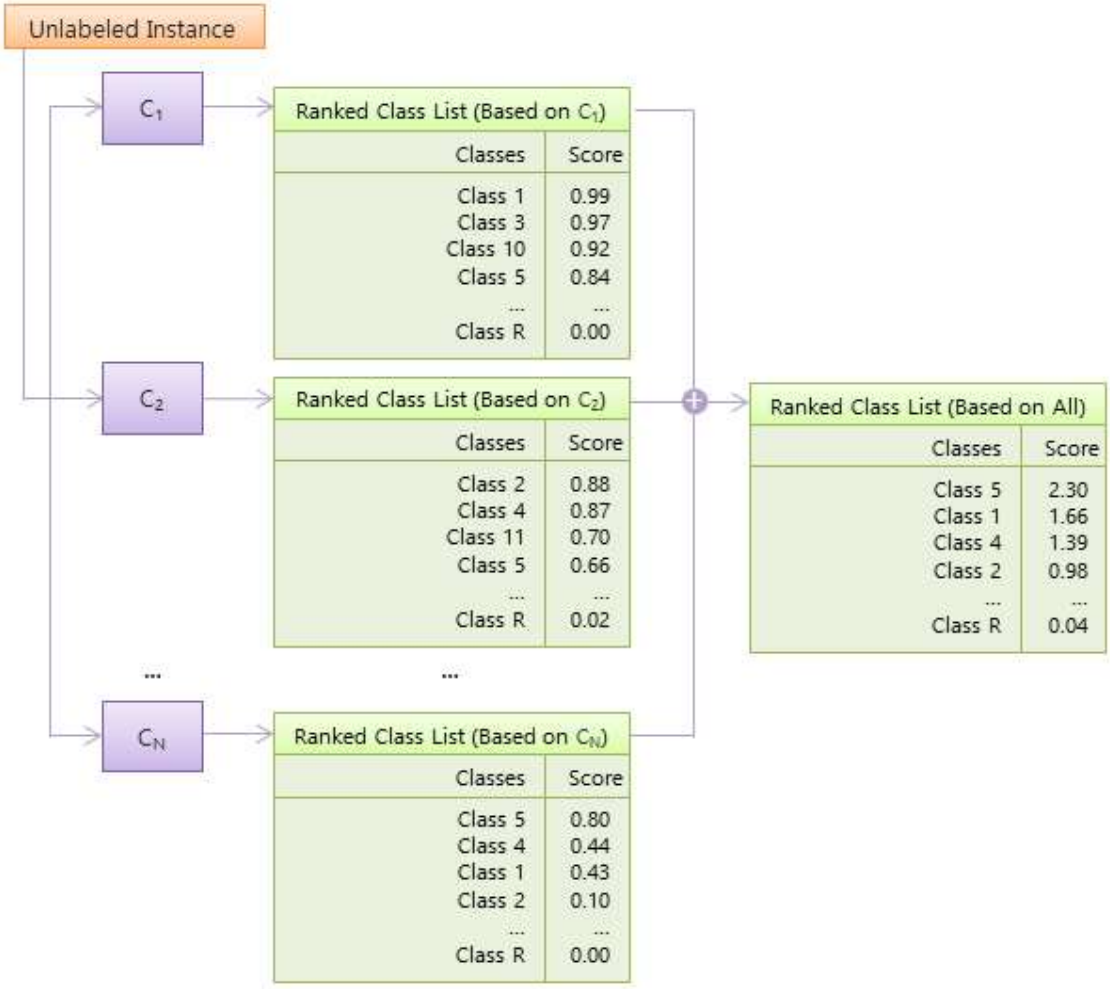


Figure 4.1. Example flow of the ensemble classification process. R is the number of possible classes, which must be the same for all classifiers. N is the number of member classifiers.

4.2.2 Pre-Filtering of classes and training data

Each attribute discovered in the raw data creates another possibility of where a bullet might belong. In essence each attribute translates to a class, and as a result the possible classes are defined directly by the raw data. The ability to adjust to different domains dynamically comes at the cost of noise, as not every retrieved attribute is a proper class. Bad attributes are not the only issue, bad values can also degrade results. To deal with this issue three filters have been developed that remove invalid classes and trim noisy data.

The first filter, presented in Algorithm 4.1, removes classes with occurrence below a certain threshold. This mainly removes classes that are most commonly found under a different name. It also removes classes that are misspelled, or that are made ad-hoc for a specific product. Some examples of such cases are "(general) sise" that is misspelled, or "(sensor) flash" that is incorrectly placed under the "sensor" category. Because such errors

are not common the resulting attributes are encountered very few times in the raw data. This filter greatly reduces the number of possible outcome classes making the classification task easier and increasing accuracy. At the same time however, accuracy might go down if proper classes are removed, since some texts might no longer have a correct class to be classified to. The nature of the data and the problem dictate if this filter should be applied or not, and to what extent. In this context the algorithm is looking for the most important attributes. By definition these should have a significant occurrence between product pages, so the rare occurrences can be safely removed.

Algorithm 4.1. Removal of rare classes

```
foreach (attribute){
    found = TimesDiscoveredInData(attribute);
    if(found / totalNumberOfAttributes < a){
        MarkAsRare(attribute);
    }
}
```

Where `a` is a properly selected constant depending on project requirements, set at 0.01 for this ensemble

The second filter is looking for outliers in the values, removing values particularly long in comparison to the average value length of that attribute. As shown in Algorithm 4.2, an interquartile range test is used to define many outliers in one pass (Natrella, 2010). The main values filtered are ones that contain explanatory texts. For example the class "(features) built-in flash" has typical values of yes or no. If one value is "yes, guide number of 12m @ 100ISO", then it is removed from the training data. As outliers are defined in contrast to the other values, removing them and repeating the procedure might discover new outliers. This also allows the filter adapt to the given data.

Algorithm 4.2. Outlier detection using Interquartile Range Test.

```
foreach (attribute){
    q1 = Q1(length_of_values);
    q3 = Q3(length_of_values);
    iqr = q3 - q1;
    min = q1 - (iqr * a);
    max = q3 + (iqr * a);
    foreach(value){
        if(value_length < min OR value_length > max )
            MarkAsOutlier(value);
    }
}
```

Functions `Q1()` and `Q3()` find the value representing the middle of the lower half and upper half of the data respectively. `a` is a properly constant that makes the test more or less aggressive, typically 1.5.

The final filter transforms or erases classes with an average value length over a given threshold. Initially the filter attempts to identify if any of the values can be split into multiple smaller parts. If a pattern is found then the value is split and each part is regarded as a separate value. For example, the class "(sensor) ISO modes" mainly contains values that list all modes that a camera supports (e.g. ISO 100/ ISO 200/ ISO 400/ ISO 800). These are split into individual pieces and treated as separate values. When all patterns have been identified the average length is examined, and if over the threshold the attribute is removed. A split pattern is assumed, if the lengths of the resulting parts are within 10% of each other (or a minimum of 3 characters). Algorithm 4.3 summarises this process. This filter is effective because by definition bullets are small texts, so such particularly long values are almost never encountered. The application of this filter also depends on the nature of the problem. If the problem is occupied with identifying only a specific type of classes, then the rest can be removed to improve accuracy. For the given context classes with long and textual values are not relevant, and as a result this filter can be used. Bullets are also passed through this filter before they are classified.

Algorithm 4.3. Removal of long attributes

```
foreach (attribute){
     foreach (value){
         parts = Split(value);
         attribute.removeValue(value);
         attribute.addValues(parts);
      }
   avg = Average(length_of_values);
   if(avg > a){
      MarkAsTooLong(attribute);
   }
}
```

`Split()` is the function that implements the pattern detection. `a` is a properly selected constant depending on project requirements, set at 40 for this ensemble

4.2.3 The member classifiers

Each classifier uses completely different criteria to determine to which attribute a bullet might belong. For the following discussion the examples are pulled mainly from Digital Cameras. This is not because the classifiers are built specifically for cameras, but to help the explanation of the feature sets trough comparable examples of bullets and attributes.

4.2.3.1 Name Classifier (NC)

The Name Classifier (NC) matches bullets that either consist of or contain the attribute's name. This is achieved by looking for direct mentions of the attribute's name in the bullet.

To create the feature set of NC the attribute's names and categories are stemmed. Common stopwords are removed[6] and only words with 4 or more characters are considered. Each stem is used to create a boolean feature for the classifier in the form of "contains_stem", e.g. "contains_optic"," contains_memor" and so on. As a result the classifier contains a variable number of features depending on the stems extracted from the training data. This automatically adapts to the given domain with a very relevant feature set. The generic form of the feature of NC is

$$F_{NC[1\text{-}N]} = (\text{contains\_stem}_1, \text{contains\_stem}_2, \ldots, \text{contants\_stem}_N)$$

where N is the total number of stems.

For example, the attributes "(sensor) resolution" and "(general) zoom" will give the stems: sensor, resolut, gener and zoom. The feature set of an NC classifier based on those 2 attributes would be (contains_sensor, contains_resolute, contains_gener, contains_zoom). Based on that feature set bullets are translated to signatures. For example the bullet "5x optical zoom" would create the signature (false, false, false, true), since it only contains the word zoom. The bullet "12MP" would create the signature (false, false, false, false) since it does not contain any of the words.

In the lack of already classified bullets, NC is trained using the names and categories of attributes. This is achieved by treating the class names themselves as bullets. To continue on the previous example, the attribute "(sensor) resolution" would create a signature of (true, true, false, false). It is already known to which attribute this signature refers, and as a result this can be used as a training instance. Essentially, NC is being taught that if a bullet contains any words of the class name itself, then it belongs to that class. Each attribute creates two training instances, one from the name only and one from the combination of the name and category. Table 4.1 shows an example of the training instances created from the attributes "(sensor) resolution" and "(general) zoom". The training instances are translated to their respective signatures based on the features of NC, as per the last examples. Using the category name is crucial, as it allows it to identify bullets that need to mention the category in order to be meaningful. For example, "3cm sensor size" and "3cm display size" ought to mention their respective categories (i.e. sensor and display) so as not to be confused with

[6]Based on a list common stopwords, as found at https://github.com/arc12/Text-Mining-Weak-Signals/wiki/Standard-set-of-english-stopwords.

each other. Other attributes however are not logically obliged to mention the category, which is why both cases are covered in the training set.

Table 4.1. Creation of training instances for the NC with the feature set (contains_sensor, contains_resolute, contains_gener, contains_zoom)

Part of attribute used	Signature		Class
resolution	(false, true, false, false)	→	(sensor) resolution
(sensor) resolution	(true, true, false, false)	→	(sensor) resolution
zoom	(false, false, false, true)	→	(display) size
(general) zoom	(false, false, true, true)	→	(display) size

Since there are only two different training instances for each class there are no underlying patterns discover, and as a result NC is implemented very efficiently with a Naive Bayes classifier. Bullets that do not contain any words from the attribute's names, will all have signatures filled only with "false". For example the bullet "12 MP" and the bullet "Waterproof" will both have a signature of (false, false, false, false). Since they have the same signature, from the perspective of NC all these bullets look identical and would be classified in the same class. In addition, since by definition there is not one single training instance with that signature, NC does not have any training on where to classify those bullets and the result would be arbitrary. For those reasons, the results of NC are completely negated and not used for the ensemble when such cases occur.

4.2.3.2 Value Classifier (VC)

The Value based classifier (VC) matches bullets that have similarities with the values of the attributes rather than the names. The feature set of VC has a variance of different features, logically separated in 3 groups. The first set of features attempts to match letter and symbol patterns, and it consists of finding the presence for each of the following 43 characters,

abcdefghijklmnopqrstuvwxyz<>,.?;:'&()/*-+="

This produces Boolean features such as "contains_a"," contains_b", etc. The occurrence of capital letters is excluded because they create false correlations. The SWC (whose discussion follows) is instead responsible to capture these letters. The first 43 features from VC's feature set are the following

$$F_{VC[1\text{-}43]} = \textbf{(contains\_a, contains\_b, …, contains\_z, contains\_<, contains\_>, …, contains\_")}$$

The next 5 features capture meta-data about the bullet. These are the number of digits, the number of integers (as full non-decimal numbers, not as a one by one digit), the number of words, the total number of upper-case letters, and the total length of the value. As a result, features 44 to 48 are the following

$$\mathbf{F_{VC[44\text{-}48]} = (numOf\_digits, numOf\_integers, numOf\_words, numOf\_uppercase, numOf\_characters)}$$

The last group of features attempts to identify numbers that exist into certain number ranges (bands). This creates numerical features in the form of "numbersInBand_[12.1-18.4]", which measure how many numbers within this range exist in a bullet. The amount and the boundaries of the bands can vary based on the training data. The remaining features are thus in the following form

$$\mathbf{F_{VC[49\text{-}N]} = (numbersIn\_[b_{1m}\text{-}b_{1M}], numbersIn\_(b_{2m}\text{-}b_{2M}], \ldots, numbersIn\ \_(b_{nm}\text{-}b_{nM}])}$$

where m and M are the min and Max of each band respectively, n is the total number of created bands, and N is the total number of features (N = 48+n).

In essence, because numbers are infinite and impossible to represent as distinct features, they are grouped into bands. This process is called discretization and is a form of dimensionality reduction (H. Liu & Motoda, 1998). A number of discretization techniques can be used, such as K-means clustering, equal width binning or equal frequency binning (Catlett, 1991; Kerber, 1992). VC is implemented with equal frequency binning, while the details of how equal frequency binning works have been discussed in section 3.4.4, step 5a, p.37. To create the bin ranges the unique occurrence of each number found in the attributes is being used, as well as a maximum of 60 bins or a minimum of 3 values per bin. As a result of the above, VC has a variable number of features between 48 and 108.

Each attribute, has one or more values as these have been gathered by the web crawler. For example the attribute "(sensor) resolution" may have the following values "12.1 MP, 14.5 Megapixels, 21.4 MP, 18MP". Each of these values can be used to train VC, and as a result VC will be capable to recognize such values in bullets. For example, assume a VC with the following simplified feature set (contains_a, numOf_digits, numbersIn_[12.1-18.4]). Table 4.2 shows the creation of some training instances from a few values of a few attributes.

Table 4.2. Creation of training instances for the VC with the feature set (contains_a, numOf_digits, numbersIn_[12.1-18.4])

Value	Signature		Class
12.1 MP	(false, 3, 1)	→	(sensor) resolution
14.5 Megapixels	(true, 3, 1)	→	(sensor) resolution
21.4 MP	(false, 3, 0)	→	(sensor) resolution
18MP	(false, 2, 0)	→	(sensor) resolution
3 inch	(false, 1, 0)	→	(display) size
3.2 inch	(false, 2, 0)	→	(display) size

Because discretization is based on the training data, VC can adjust very well on the numbers appearing in the context that it is applied. Using bands of numbers allows capturing of numbers of similar magnitude even if those appeared very rarely in the training data, or did not appear at all. To illustrate, VC is able to identify a relation between "16 Megapixels" and "15.9 Megapixels" even if these values were never present in the training data (assuming both these numbers end up in the same bin). The use of letters as features creates strong connections between them, practically allowing the classifier to detect words. For this reason, VC is implemented by a Multilayer Perceptron (MLP).

4.2.3.3 Units Classifier (UC)

The Units Classifier (UC) finds the units in which an attribute is measured. This is achieved by getting the next word or symbol after a number, if any number is present, regardless of whether it is an actual unit or not. A list with all the possible different units for the domain is populated by scanning all the values of all attributes. The nominal feature "measured_in" is then constructed using the contents of the unit list as possible options (e.g. measured_in=[cm, hours, megapixel]). The unit list contains very little noise which is almost limited to number delimiters such as dashes, slashes or commas. Without loss of generality, these common elements can be removed to reduce the chance of false positives:

-*/?;,.()[]{}\|~

A dictionary is used to combine different expressions of the same unit (e.g. double quotes with inch and inches, centimetre with cm and so on). Two more options are always added to the list of units; one for the case where no units are present, and one for the case where units are present but UC is not aware of them. The latter happens when a unit did not exist in the training data, and in essence this option represents "all other units". In case more than one units are present only the latest one is used. A second feature is used to detect if an attribute

is boolean (i.e. yes/no or true/false) or not. As a result, the final feature set of UC contains the following 2 features:

$$F_{UC[1\text{-}2]} = \textbf{(measured\_in=}[unit_1, unit_2, \ldots, unit_N, noUnit, anyOtherUnit]\textbf{, isBoolean)}$$

where N is the total number of units discovered from the attributes' values.

Similar to VC, UC is trained using the values of all attributes as bullets. Table 4.3 shows an example where UC is created from the 6 values of 4 attributes. Collectively these 6 values contain 3 units, MP, Megapixels, and inch.

Table 4.3. Creation of training instances for the UC with the feature set (measured_in=[MP, Megapixels, inch, noUnit, anyOtherUnit], isBoolean)

Value	Signature		Class
12.1 MP	(MP, false)	→	(sensor) resolution
14.5 Megapixels	(Megapixels, false)	→	(sensor) resolution
Yes	(noUnit, true)	→	(other) waterproof
Carrying Strap	(noUnit, false)	→	(other) extras
3 inch	(inch, false)	→	(display) size
3.2 inch	(inch, false)	→	(display) size

If a bullet does not contain any units and is not boolean, the results of UC are negated during the combination. This is because too many classes do not contain any units at all, and as far as UC can detect they all look alike (since their signatures are the same). If a bullet contains a unit that was not seen in the training set, UC does not have any information on where to classify it, and the results are also negated in such cases. This is also the main reason UC needs to be separated from VC, highlighting once again the power of having multiple separate classifiers.

Because the list of units is populated by the training data, it can adapt to the given context automatically. The predicting power of UC lies in the fact that generally, no more than a couple of classes will use the same units. However, once limited to these few classes there is no way of knowing which one is correct. As the decision is based on two mutually exclusive features (it either contains a unit, or is boolean), a Naive Bayes classifier is very effective.

4.2.3.4 Special Words Classifier (SWC)

The Special Words Classifier is used to detect the special words and acronyms often used in product descriptions. In essence, it can differentiate between the normal and the possibly special use of letters and numbers. The number of each of the following characters is used to create the feature set

[A-Z][a-z][0-9][.-]

As a result, the feature set of SWC contains the following 38 features

$$F_{SWC[1\text{-}38]} = (numOf\_A, numOf\_B, \ldots, numOf\_., numOf\_-)$$

where each character is counted only where special use if assumed.

To determine which parts of a value are considered "special" the following steps are followed.

1. All bracket symbols are removed, i.e. (), [] and {}.
2. Values are split in white-spaces and each segment is treated separately.
3. Single symbols at the start or end of the segment are removed.
4. Special use is assumed, unless the segment falls in any of the following categories.
5. Segments with only 1 character.
6. Segments that are a numbers (integer or decimals).
7. Segments that are a number with units (e.g. 9600p).
8. Segments in the "something-by-something" format, with 2 or more elements, regardless of units in the end (e.g. 100x20x20, 1536x9600p).
9. Segments that contain only letters, and all letters after the first one are lowercase (e.g. optical, Optical)

Table 4.4 presents some examples of where special words are identified.

Table 4.4. Example cases where special character meaning is assumed

Value	Segments	Reason to consider normal
Movie recording (H.234)	Movie recording **H.234**	Case 9 Case 9 **SPECIAL**
Triple XD Engine	Triple **XD** Engine	Case 9 **SPECIAL** Case 9
250 cd/m2	250 **cd/m2**	Case 6 **SPECIAL**

Conventional 4:3	Conventional **4:3**	Case 9 **SPECIAL**
2ms gray-to-gray	2ms **gray-to-gray**	Case 7 **SPECIAL**
1 x HDMI (ARC)	1 X **HDMI** **ARC**	Case 5 Case 5 **SPECIAL** **SPECIAL**

Like the last two classifiers, SWC is trained using all the values from all attributes. Values are first filtered through the above algorithm. The signature is created based only on the segments identified as special, if any. If no special words are found, then the training instance is not used at all. Table 4.5 contains some training examples, for a demo SWC with a reduced feature set.

Table 4.5. Creation of training instances for the SWC with the feature set (numOf_A, numOf_B, numOf_1, numOf_2, numOf_.)

Value	**Special Words**	**Signature**		**Class**
Raw, A.112	A.112	(1, 0, 2, 1, 1)	→	(Video) encoders
Triple AB Engine	AB	(1, 1, 0, 0, 0)	→	(Other) Image enhance
Conventional 1:2	1:2	(0, 0, 1, 1, 0)	→	(Image) format
3.2 inch		DISCARDED		

If no special segments are found in a bullet, then the signature will contain only zeros. This means that from the point of view of SWC, all these bullets are the same, while in truth their only similarity is that they contain no special words. In addition, since we discarded training instances with no special words, SWC has no information on where to classify such bullets. To avoid false correlations, the results of SWC are negated for the ensemble in such cases.

SWC is particularly effective in capturing versions, models, technology acronyms etc. The classifier could be used as-is in any context that might contain similarly modelled special words, though in their absence its effectiveness is diminished. SWC also demonstrates how to utilize characteristics of the text that are relevant specifically to the given domain. For this classifier a Multilayer Perceptron was preferred because the connection between features can actually identify whole words rather than individual characters.

4.2.4 The ensemble

When a bullet is classified, each classifier will produce a probability score for each attribute. The scores are added together to produce one final score for the whole ensemble. The scoring function is $f(x) = \sum_{n=1}^{N}(s_n w_n)$, where N is the number of classifiers used, s_n is the score produced by classifier n, and w_n is the weight assigned to classifier n. In the example of Figure 4.1 the results of each classifier are just summed together (which equates to all weights being 1). Table 4.6 presents a more accurate example by using the actual weights of the actual classifiers. In this example the bullet "24 Megapixels" is being classified to 3 possible classes, i.e. zoom, resolution, and flash type. "Score" is the raw score as it is produced from each classifier, while "W. Score" is the Score multiplied by the "Weight". The weighted scores are then added for the Ensemble and the class with the highest score is chosen. In this example, as there are no special words identified, the results of SWC are negated by replacing its weight with 0. The ensemble would finally classify the bullet to "Resolution". It is interesting to note that the use of weighted average rather than plurality voting, allows the ensemble to select a class that might not be the most commonly selected class of all classifiers, or even a class that would have never been picked from any of the member classifiers.

Table 4.6. Example Classification of a the bullet "24 Megapixels"

Classifier	Weight	Classes (Attributes)					
		Optical Zoom		Resolution		Flash Type	
		Score	W. Score	Score	W. Score	Score	W. Score
NC	6.50	**0.5**	3.25	0.1	0.65	0.3	1.95
VC	0.20	1.3	0.26	**5.4**	1.08	1.5	0.30
UC	2.25	0.5	1.50	**2.0**	4.50	0.1	0.23
SWC	0.00	0.4	0.00	0.3	0.00	**0.9**	0.00
Ensemble			5.01		**6.23**		2.48

Highlighted in bold are the highest scores of each classifier, representing the choice each classifier would make if working on its own.

In essence, much like the negation of results, weights are a form of knowledge pre-built into the system. The final weights used are the following:

NC: 6.50 VC: 0.20 UC: 2.25 SWC: 1.05

It should be noted that different classifiers give results of different magnitudes, and thus the weights cannot be used directly as an indicator of the importance of each classifier.

The weights are set empirically to specifically match this set of classifiers and consumer electronics. This was based on a separate subset of only 30 bullets that were annotated by hand and used to evaluate the effectiveness of the ensemble. To get the weights, all possible combinations of weights were tested by moving each weight between 0 and 10 in 0.05 intervals, for a total of 1340283 permutations. Algorithm 4.4 is used to perform that test. The combination that produces the highest accuracy is chosen.

Algorithm 4.4. Selecting the best weights

```
accuracies = new array[][][][]; //stores the accuracy of each combination
for(double nc = 10;  nc>=0;  nc-=0.05){
  for(double vc = 10-nc;  vc>=0;  vc-=0.05){
    for(double uc = 10-nc-vc;  uc>=0;  uc-=0.05){
       double swc = 10-nc-vc-uc;
      double accuracy = ClassifyEvaluationSet(nc, vc, uc, swc);
      accuracies[nc, vc, uc, swc] = accuracy;
    }
  }
}

bestWeightSet = GetKeysetWhereMax(accuracies);
```

`ClassifyEvaluationSet()` classifies the whole evaluation set and returns the accuracy using the given set of weights. `GetKeysetWhereMax(array)` returns the set of keys where the highest accuracy was found.

The selection of weights is a great candidate for future work, as its automation would completely eliminate the need to hand tag data. In his work, Jiménez (1998) has discussed a methodology that uses the confidence level of each classifier as a weight. This means that weights are not only produced automatically, but are adjusted on a per-classification level, rather than once per system. However, because this ensemble is created with fundamentally different classifiers, his work is not directly applicable.

4.3 Evaluation

This section evaluates the ensemble classification methodology and its suitability to solve the bullet classification issue. The following sections explain the evaluation procedure, and discuss the performance of this module as a standalone component, rather than as a part of the greater recommender system.

4.3.1 Set-up

To evaluate the presented methodology a demonstration system was developed in Java. The classifiers were implemented using the well-known WEKA framework for machine learning and data mining (Hall et al., 2009). Two different datasets were used for evaluation, i.e. Cameras and TVs. Data were obtained using live commercial websites, namely Argos (190 products), Curries (30), Jessops (116) and Tesco (407) for Cameras, and Curries (207), Pixmania (225) and Tesco (300) for TVs. All available products were retrieved from each web site[7]. The webpages were extracted using the crawler and custom content extractor described in section 3.2. Specifically, the extractor identifies the bullets, as well as the category headings and pairs of the attributes' name and value. Table 4.7 presents more information about the created datasets. Each of the collected products has a number of attributes, many of them repeating between products. These attributes are the possible options of where a bullet might belong, and ultimately form the classes of the classifier. Each of these attributes has one or more values, the number of values of all attributes is also shown on the table. Finally, the number of Bullets is how many bullets have been extracted collectively from all product webpages, while Unique Bullets shows how many unique literal strings have been found.

Table 4.7. Evaluation Datasets

Set	Products	Attributes	Total Values	Bullets	Unique Bullets
Cameras	743	297	18799	930	327
TV	732	420	21259	1659	202

All information is before any filtering takes place.

In regards to the pre-filtering of bullets, two iterations were found more than adequate for outlier removal, with the first iteration alone removing 96% of total values removed. A third iteration does not remove any additional values. More than 80% of the retrieved bullets have a length of less than 20 characters, for a total average length of 17 (before any filtering).

[7] Webpages retrieved 15/10/2013

This further supports the use of the filter that removes attributes whose values are consistently too long (i.e. text). As the maximum average length allowed is reduced, accuracy increases since there are less classes to choose from. However, the reduction in length increases the risk for erroneously discarded classes and bullets. We've set the maximum allowed length to 40, which permits ~95% of bullets to go through[8].

Since training and evaluation data are in a different formats, overfitting is not directly possible. For the same reason, the widely used K-fold cross validation test cannot be applied, as separating a portion of the bullets does not provide proper training data. Instead, we use a portion P of randomly selected webpages to provide the training data, and all the bullets are always used for evaluation. We run tests for P values of 5%, 10%, 20%, 30%, 40% and 50%. We repeat each test 30 times, and present the averages.

Tables 4.8 and 4.9 show the training data gathered for each P, for Cameras and TVs respectively. For Cameras, 149 attributes, roughly 50% of the total (297), can be discovered from just 5% of the products, rising to 95% for 50% of the products. TVs follow a similar trend with 60% for P=5% and 90% for P=50% (420 maximum attributes). This is easily explained as attributes repeat often between webpages. Even though the effects of the filtering mechanism have not been tested separately, it is interesting to note that as P increases filtering is able to identify and discard not just more attributes, but a bigger portion of the attributes. This supports the previous explanation that important attributes are discovered early. It further suggests that with more attributes the mechanism adapts and the perception of what is important changes.

Table 4.8. Training Data Gathered for the Camera Dataset

Percentage of products used for training (P)	Attributes Before Filtering	Total Values Before Filtering	Attributes After Filtering	Total Values After Filtering	Attribute Reduction	Values Reduction
5%	149	967	130	954	13%	1%
10%	193	1845	157	1818	19%	1%
20%	222	3752	168	3674	24%	2%
30%	244	5596	176	5463	28%	2%
40%	260	7476	183	7302	30%	2%
50%	274	9620	182	9361	33%	3%

[8] Presented numbers are averaged over both datasets.

Table 4.9. Training Data Gathered for the TV Dataset

Percentage of products used for training (P)	Attributes Before Filtering	Total Values Before Filtering	Attributes After Filtering	Total Values After Filtering	Attribute Reduction	Values Reduction
5%	253	1082	241	1028	5%	5%
10%	293	2085	279	2016	5%	3%
20%	328	4389	312	4320	5%	2%
30%	354	6489	334	6439	6%	1%
40%	370	8718	350	8646	6%	1%
50%	379	10667	356	10505	6%	2%

For the purposes of evaluation three lists are manually created.

- ListA notes the correct classes for each bullet. Bullets might have more than one correct class as they might directly refer to more than one attributes. For example, the bullet "3inch LCD screen" contains both "(screen) size" and "(screen) type".
- ListB groups classes that refer to the same attribute but have different names. For example "(screen) type" and "(viewfinder) type". The criteria for this annotation was that the values of the attribute must be in a similar format. For example the classes "(flash) type" and "(other) flash" are not merged, as the typical values of the first are "built-in, pop-up, etc.", and the values of the second are "yes/no".
- ListC holds a directed graph of classes that contain other classes. For example "(sensor) type and size" contains the distinct classes of "(sensor) type" and "(sensor) size". The criteria for this list is that the parent class should offer at least all the information contained in any given child.

4.3.2 Classification Accuracy

To measure the accuracy of the classification ListA is expanded to include all similar classes from ListB. If the classification result exists in this expanded ListA, it is counted as correct. To continue the previous examples, if "3inch LCD screen" is classified to either "(screen) size", "(screen) type" or "(viewfinder) type", it is counted as correct.

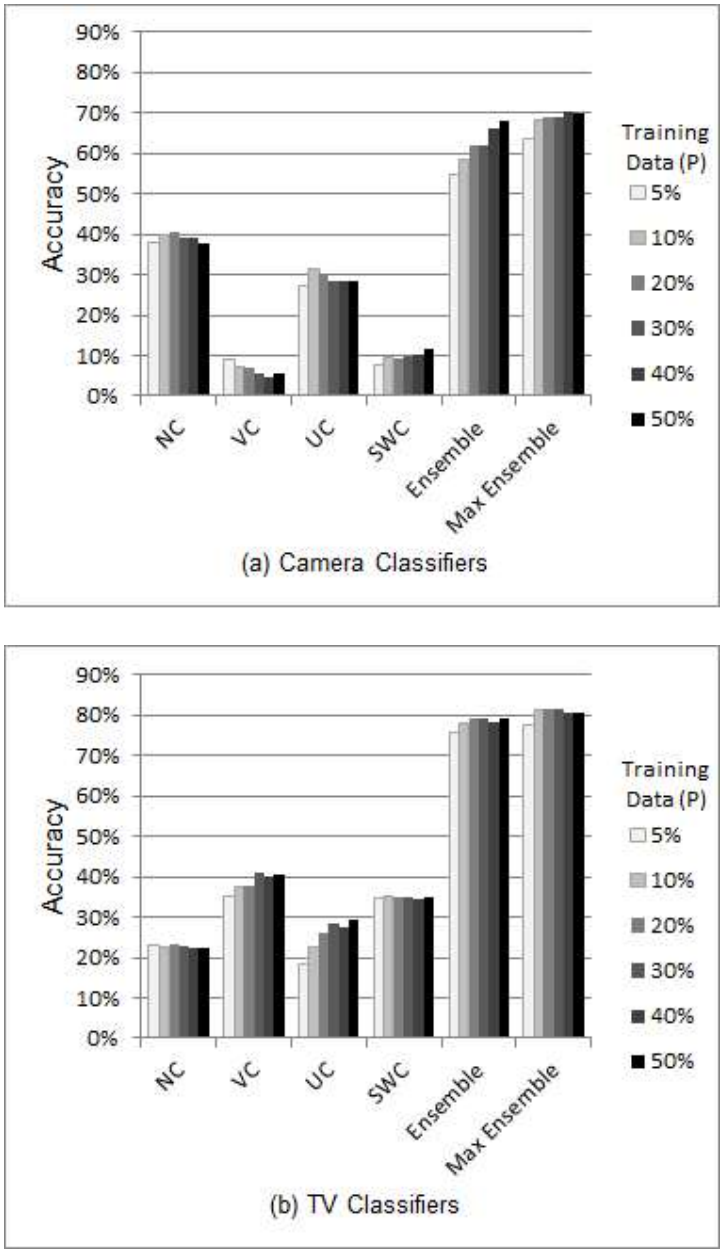


Figure 4.2. Correct predictions of each classifier for the Camera dataset (top) and TV dataset (bottom), for different percentages of training data (P).

Figure 4.2 summarizes the results of 30 iterations for different amounts of training data (P). For example in Figure 4.2(a), the first bar from the left shows that when using a random 5% of the product webpages as training data (P=5%), the Name Classifier (NC) alone can (on average over 30 repetitions with a different training set) correctly classify 38% of the bullets. It is immediately obvious that the ensemble has consistently better accuracy than any standalone classifier. Specifically, the improvement over the best performing classifier ranges

from 144% to 168% for Cameras, and from 195% to 214%% for TVs[9], depending on P. It is important to note that the Camera dataset has a baseline error rate of 16% due to noisy bullets with no respective classes (for example promoting texts such as "amazing offer"), capping the maximum possible correct classifications to 84%. The equivalent number for TVs is only 2%[10].

Max Ensemble represents the potentially correct classifications of the ensemble, if the weights of the classifiers were readjusted per iteration to achieve the maximum possible correct classifications. To clarify how this number is produced, Algorithm 4.4 from section 4.2.4 is used at each iteration to select the best performing set of weights. The result presented in the graph is the average of the best possible combination per iteration. As expected, the fixed weights do not give the best possible results in every iteration, but are consistently close, and get even closer with more training data. This demonstrated how a fixed weighted scheme is an appropriate approach to combine the results of a non-dynamically created ensemble.

Another interesting observation for the Camera dataset is that even though 3 out of 4 classifiers tend to degrade their results as P increases, the ensemble consistently improves. This is made possible by using weighted average rather than plurality voting, which gives to the ensemble the ability to consider all the results, and not just the top picks of every classifier (example in Table 4.6). Consequently, the ensemble can make a more informed decision, which as it can be seen in the results is correct more often than individual classifiers.

Since there are no other existing methodologies that we can use to utilise our training data, there is no baseline for comparison and results need to be judged per se. These results are acceptable for the intended use, and promising enough for the application of the methodology in different contexts. However, to further explore this finding the following test is also performed.

4.3.3 Method effectiveness

Apart from the classification accuracy, the effectiveness of the whole procedure is also evaluated in context. To achieve this, we measure the existence of the top 7 attributes identified as important, in the list of the top 7 actually important attributes.

[9] Due to constraints in processing power, the MLPs of VC and SWC were replaced with Naive Bayesian classifiers for the TV dataset only.

[10] 149 bullets out of a total of 930 for cameras. 29 out of for 1659 TVs.

The list of actually important attributes occurs by measuring the repetition of each correct class from ListA. Utilizing ListB similar classes are merged. Finally, because the bullets that refer to a child class will refer to the parent class as well, parent classes are removed if any of their children are found in order to make room for more classes (the parent-child relationship is defined in ListC, p.63). The list of identified attributes occurs by simply summing up the results of the classification, and thus remains independent of the lists A, B and C, or any other human interaction. Both lists are finally trimmed to 7 elements. Each identified attribute is checked for existence in the list of actual attributes. Attributes might exist either directly or by any of their children. Existence is also assumed if classes are found under different names. This is because from a developer's point of view, it is not of interest with which name a class was identified, so long as the class was correct.

Tables 4.10 and 4.11 contain the average results over 30 iterations for different amounts of training data (P). With 5% of the training data used the ensemble correctly identifies 75% of the top attributes for cameras and 81% for TVs. This implies that overfitting is indeed not an issue, since accuracy is already at high with just 5% of the data used for training. Increasing P offers a double benefit, as it improves both accuracy and stability. The rate of improvement is in line with the total correct classifications previously discussed. From a different perspective the top 7 most important attributes are always included in the top 11 identified, for both sets, for every iteration. To demonstrate the impact this has on practical applications, a developer (or better yet, a crowdsourcing system) looking to identify the most important attributes, can now only examine 11 attributes rather than 297 or 420 that were originally found in each set. This is a significant reduction, which effectively makes the bullet classifier an integral component of the greater recommender system.

Table 4.10. Important Attributes Identified for the Camera dataset

Percentage of training data (P)	Found in top 7	Relative Standard Deviation
5%	5.27 (75%)	16%
10%	5.43 (78%)	13%
20%	5.63 (80%)	10%
30%	5.63 (80%)	9%
40%	5.83 (83%)	6%
50%	5.76 (82%)	8%

Table 4.11. Important Attributes Identified for the TV dataset

Percentage of training data (P)	Found in top 7	Relative Standard Deviation
5%	5.70 (81%)	14%
10%	5.70 (81%)	10%
20%	5.63 (80%)	14%
30%	5.83 (83%)	10%
40%	5.73 (82%)	8%
50%	5.83 (83%)	6%

4.4 Summary and Conclusions

This work has shown that using highly diverse classifiers creates synergy, and dramatically improves results. It has also demonstrated that separated classifiers have multiple advantages.

Each member of the ensemble can be completely different, implemented by different classification models, with different feature sets and trained with different data. Separating the classifiers allows the modification of results separately for special cases, essentially pre-encoding knowledge into the system. Some rules are used in special cases that completely negate the results of a classifier when its opinion should not be relevant. This is one of the major benefits of having distinct classifiers rather than one classifier that aggregates all the features. Another benefit is the ability to implement each classifier with the most appropriate classification model, which leads to increased accuracy and reduced training times. By keeping each classifier independent training and classification can be easily parallelised. Finally, splitting the classifiers allows for training using different training data. This makes the separation not just beneficial, but mandatory, as this is what essentially allows the ensemble to bypass the original lack of training instances.

The evaluation has shown that using a custom ensemble of classifiers is a feasible solution to the problem. The ensemble can correctly identify more bullets than each classifier would individually. This is heavily impacted by the negation of results, since the ensemble can decide without being affected by the opinions of the classifiers that do not have enough data to make a proper suggestion. The evaluation of this methodology in context provides a better understanding of its power. Results for this case scenario, show that was able to reduce the potential options by 96% while maintaining zero error rates.

In addition, the feature set of each classifier has successfully captured its respective aspect, providing some insight on what can be used as an effective feature set for product classification. Combining heuristic methods with existing classification techniques can utilize hidden characteristics of the available data. The proposed ensemble approach offers the opportunity for tuning in various stages, providing valuable flexibility. Essentially, a targeted version of the system could be developed per product category or domain.

It is important to realize that even though the needs of this work were different, the developed classifiers are able to classify arbitrary texts to explicit classes. This allows the methodology to be used for other problems, such as consolidating multiple fields of a database or multiple databases to one. The presented methodology requires human involvement limited to the tuning of parameters before executing it. Default parameters for consumer electronics have been presented in this thesis. The outcome of the ensemble can be

further improved if access to human intelligence is assumed. For example, in the case of this work the top 11 attributes can be presented to an expert so the top 7 can be selected. This is a significant improvement over presenting every attribute of the category. In the case of merging database fields, it would allow the suggestion of fields that might need to be merged.

Since there are misclassified values there is also room for improvement. Potentially, this can be achieved with better weight tuning and more heuristic procedures during the combination of the results of the member classifiers. Methodologies that use the confidence of the classifier as a starting point for the weight might be applicable. The presented methodology only identifies one attribute from each bullet. It would seem beneficial to define this mapping as one to many, to best capture the information conveyed by the bullet. The whole algorithm can be further improved with even more targeted feature sets, since the potential for hidden characteristics in values is endless. As demonstrated, classifiers can be tailored to specific problems. In the presented methodology SWC captures characters with special meaning. Consumer electronics often contain a lot of acronyms, but in their absence the effectiveness of SWC would diminish. This however points to the idea that different problems might have different characteristics that can be exploited. The approach taken allows adding or removing classifiers with virtually no changes to other parts, creating yet another adjustment point.

5 ADAPTIVE NUMBER OF ANSWERS

As it was briefly discussed in Chapter 3, crowdsourcing systems often require the input of multiple people on the same subject. That is because putting faith on the ability and the benevolence of a single, unknown and probably non-expert individual, does not form solid basis for data acquisition. The question that arises then, is how many people should be asked before a decision can be made. Choosing the wrong number of people can have negative effects on any crowdsourcing system. Much like sampling for a survey, too few might not be able to properly represent the population and too many is a waste of resources. Furthermore, requesting a fixed number of answers for all questions might also be a waste. Some questions might be easier to answer, and thus the crowd might be able to converge faster to the correct solution. On the other hand very difficult or polarized questions might need the input of more people to reach the same level of accuracy.

Not much information can be found on how to fluctuate the number of gathered answers depending on the needs of each individual question. This work proposes a methodology that can dynamically decide the required number of answers as the question progresses, effectively adapting to each question. This can reduce the average number of answers required to achieve a certain level of accuracy, or depending on the use, increase the accuracy while maintaining the same number of answers (on average). In addition, this work can be adapted to include user quality metrics which might further increase those benefits.

The Adaptive Number of Answers (ANA) methodology gauges the question's difficulty by examining the difference between the most popular answer and the second most popular answer, and stopping when a condition is met rather than when a fixed amount of answers has been gathered. The methodology is tailored for single-answer multiple-choice questions, whose results are used to determine the dominant answer rather than utilising the distribution. To better explain this with an example, assume a YES/NO question. This research examines cases where the system designer wants to find whether the answer is YES or NO, and not the distribution of answers, e.g. 70% YES, 30% NO. This question type is very common in crowdsourcing systems as there is usually one correct answer and the remainder of the population is wrong. For example, in the hypothetical character recognition question "Is this letter A", the interest typically is on whether this is indeed letter A or not, unlike a survey that would probably like to measure to what extent people think this is A or not.

We test the proposed solution by producing a table that shows the relation between the number of answers acquired and the accuracy of the result. This is achieved by simulating users answering questions in a lab environment, and measuring the ability of any number of

answers to select the correct result. Using this table as baseline, similar results are produced for ANA and the results are compared.

5.1 Fixed number of answers

5.1.1 Methodology

This work examines multiple choice questions that have only one single right answer. If we ask an infinite population about an A/B/C question, the result might be a distribution like 30% A, 20% B, 50% C, in which case a crowdsourcing system would conclude that the right answer is C. The goal of the experiment is to determine what sample size needs to be drawn from that infinite population before it can be stated that "C" is the right answer, or else that C will be indeed dominant if we were to examine the whole population.

To be able to test this without the need for real users, the infinite population previously discussed is represented by the distribution of its answers, in this example 30% A, 20% B, 50% C. Knowing the distribution, allows answers to be generated randomly but based on that distribution, through a generator that will produce 30% A, 20% B and 50% C. This generator can then replace real users, and thus allow to scale up the experiment.

To implement this functionality random answer generators like the one in Figure 5.1 are built. Internally the generator has a table of 100 slots. Each possible answer occupies one slot for each 1% it has in the distribution. To generate an answer, a dice is rolled between 1 and 100, and the respective slot is selected. The example generator from Figure 5.1 will generate "A" for dice rolls between 1 and 30, "B" for rolls 31-50, and "C" for rolls 51-100. The generator can also be queried to learn which is the correct answer (in this case "C"). The code to create such generators, along with the code for all experiments, is publically available at https://github.com/alianos-/minimumAnswers.

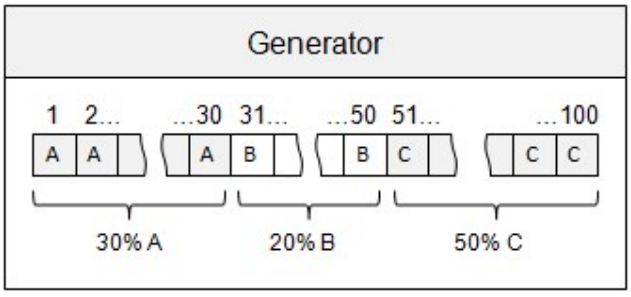


Figure 5.1. Representation of an answer generator.

Our experiment consists of creating 10 million such generators with random distributions, each representing one question. We then draw N amount of answers from each generator, and see if they were able to predict the right answer. We vary N from 1, up until

99.99% of the predictions are correct. Simple plurality voting is used for the prediction. We define accuracy (A) as the average correct predictions over 10m iterations of the above experiment. According to National Statistical Service of Australia[11], the sample size needed to produce results with a confidence level of 99% and a confidence interval of 0.0005 from a population of 1peta ($1x10^{15}$), is ~6m. We chose 10m as a number that is well over that number, and as such results are expected to be very good representations of the truth.

Distributions where there is a big difference between the most popular and the second most popular answer, represent easy questions where almost everyone is able to find the right answer (e.g. 99% A, 1% B, 0 %C). On the other hand distributions where all options are close to each other, represent difficult questions where users are confused, and torn between all options (e.g. 34% A, 33% B, 33% C). Because the distribution of each generator is random and 10m generators are used, all kinds of distributions are expected to appear in the 10m iterations of the experiment. For our tests we assume that the population can always predict the right answer by plurality voting, even by the marginal difference of 1%. We implement this by discarding distributions that do not have at least 1% difference between the most popular and second most popular option, and replacing them with random new ones.

To randomize the distribution itself Algorithm 5.1 is developed. To begin, the first option is assigned a random percentage between 0 and 100. The second option is assigned a random percentage between 0, and the remainder of 100 minus the previous rolls. The process continues for all intermediate options. The remainder percentage (if any) is assigned to the last option. For example assume the creation of a distribution with 4 options.

- A is assigned through a roll between 0 and 100, assume 30.
- B is assigned through a roll between 0 and 70 (100 - 30), assume 55.
- C is assigned through a roll between 0 and 15 (100 - 30 - 55), assume 0.
- D is the final option and is given any percentage left, in this case 15 (100-30-55-0).

[11] http://www.nss.gov.au/nss/home.nsf/pages/Sample+size+calculator

Algorithm 5.1. Randomizing the distribution for a given number of options.

```
int usedPercentage = 0;
//assign percentages to all options but the last
for( int i = 1; i < numOfOptions; i++ ) {
      int remainingPercentage = 100 - usedPercentage;
      int percentage = randomRoll( 0, remainingPercentage);
      option[i] = percentage;
      usedPercentage += percentage;
}
//Assign the remaining percentage to the last option
option[numOfOptions] = 100 - usedPercentage;
```

5.1.2 Results and discussion

We repeat the described experiment for questions that have between 2 and 6 different options (O) (i.e. A/B, A/B/C, ..., A/B/C/D/E/F). For each O, we repeat the experiment with increasing N and note the achieved accuracy (rounded at 2 decimal digits). Table 5.1 shows the first N that was able to achieve each milestone accuracy (A), while the detailed results of all tests that have been performed are publically accessible at http://alianos.website/r?c=akma. To clarify, each cell in Table 5.1 is the result of 10m iterations of the experiment described in the previous section, for the given N and O. In case of a draw the result it counted as wrong, regardless of whether the correct option was among the winning options or not. For example in the draw 5A, 5B, 3C, the result is automatically counted as wrong regardless if either A or B was the correct result.

It is important to mention that Accuracy as a function of N is not monotonic, and occasionally it might drop even if N increases. This occurs mainly because an even number of answers creates more chances for a draw. This phenomenon is intense for small N or O, while it quickly fades away as either N or O increase. Even in the summarised Table 5.1 where intermediate results are missing, it can be observed that most of the small numbers are odd. Due to that phenomenon, it is not always safe to pick a bigger number to get better accuracy. As such, future system developers would be advised to look at the full table of results to make better decisions, and if not anything else, use an odd N for the binomial distribution or when N is less than 10. Of course this is only relevant if simple plurality voting is used to obtain the correct result, and if N cannot be adjusted in case of a draw.

Table 5.1. Each cell shows the minimum amount of answers (N) that was able to achieve the shown accuracy. The number in each cell, is the result of selecting that number of answers, from 10m random answer generators.

Accuracy (A)	Number of Options (O)				
	2	3	4	5	6
80%	3	10	11	11	11
85%	7	17	19	19	19
90%	13	35	39	39	39
91%	17	42	47	47	47
92%	21	51	59	59	57
93%	27	66	75	74	73
94%	37	87	97	97	96
95%	53	121	135	135	132
96%	79	179	201	199	196
97%	131	310	331	328	321
98%	310	587	648	640	628
99%	739	1735	1870	1842	1801

Table 5.1 reads as follows. For a series of questions with a given O, in order to predict the right answer in A per cent of the questions, at least N answers per question need to be obtained. For example, in a series of binomial questions (O=2), in order to receive the right answer in 95% of the questions, 53 answers per question need to be gathered. That is of course assuming that the distribution of answers can freely be anything, and questions are not consistently easy or hard.

In the 10m generators created for each N exist all kind of questions; from the easiest question possible that almost every user answers correctly, to the very hard ones where users are torn almost equally between options. As a result, Table 5.1 is a rather strict representation of real systems and thus produces rather unrealistic results. Most problems suitable for crowdsourcing should offer questions that are easier than average, since a group of non-experts should be able to complete the task. This is one of the reasons why existing systems use numbers that are drastically smaller than our findings and still get accurate results. In an opposite scenario, it would be more sensible to defer hard questions to expert review, rather than trying to achieve a uniformly high accuracy with crowdsourcing.

The methodology of this experiment is easily reproducible for a more sensible range of questions difficulty, by limiting the range of distributions that the answer generators can have. As an example, the full table was calculated for questions where the first and second most popular options of the distribution have at least 30% difference, and no more than 80%

(e.g. for O=2, distributions can range from A=90% B=10% to A=65% B=35%). The reduction in answers required is dramatic; 99% accuracy can be achieved for the binomial question with only 22 answers, down from 739 that we find in Table 5.1.

As a result of the above, it becomes safe to state that the difficulty of the question plays a major role in the final accuracy of the system. However, as it is not always easy to know or measure the average difficulty of the questions, it would make sense to avoid using a fixed number of answers system-wide, and turn to methods that can adapt on the difficulty of individual questions.

5.2 Adaptive number of answers (ANA)

5.2.1 Methodology

As it was discussed in the previous section, using a fixed number of answers for each question is not a very good approach, as it is rather difficult to decide what number should be used. An additional problem is that since different questions have different difficulty levels, the fixed number affects them differently. As a result, since the difficulty of the question is not known a-priori, one cannot be sure of the resulting accuracy level. This can have adverse effects on the system, as developers will either have to overcompensate by requiring more answers than they really need (so the hard questions can achieve acceptable accuracy levels), or require less answers and risk inaccurate results.

ANA attempts to address this issue by adapting the number of answers based on the difficulty of the question. We define difficulty in direct relation to the distribution of answers; the smaller the difference between the first and second option, the greater the difficulty. The proposed methodology suggests examining this difference with every new answer that comes in, and thus gauging the difficulty of the question on-the-fly. With every new answer that is received, it is examined if the most voted option thus far has received at least "*c*" more votes than the second most popular option, where *c* is an appropriately selected constant. If it has, the process can stop, and the most popular option is regarded as the correct one. If it hasn't, one more answer is requested, and the process continues until the condition is met.

For example, assume that a question with 3 options has received the following answers: 4A, 7B, 1C. For a constant of c=4, the methodology states that the most popular option, B, should have at least 4 more votes than the second most popular option, A. As the condition is not met one more answer is required. If the answer is another B the condition is met, so the process can stop and assume B as the correct answer. If it not, the process continues by requesting yet another answer until the condition is met. Once again, simple plurality voting is used to decide on the wining answer.

In practice that means that easy questions will grow this difference fast and the procedure can stop early. More difficult questions will require more total votes in order to reach the same distance. Another benefit from this condition is that it can be made more or less strict by increasing or decreasing the constant *c*. This can be exploited to relate the average number of answers required to certain levels of accuracy. This is determined experimentally by reproducing the experiment of the previous section using the new stop condition.

5.2.2 Results and Discussion

To evaluate ANA we modify the experiment from Section 5.1. Instead of repeating the sampling process 10m times for a fixed number of answers (N), we stop each repetition when the condition is met. We then note the average number of answers (N_A) used and the achieved accuracy (A) (rounded at 2 decimal digits). We repeat the experiment for constants, *c*, between 2 and 35, and expect that as *c* increases, the achieved A and N_A will also increase. To produce Table 5.2 we select the minimum *c* that was able to achieve the presented milestones in Accuracy (A). The whole process is repeated for questions that have between 2 and 6 possible options (O). The detailed results of the experiment can be found at http://alianos.website/r?c=akma. Similar to the previous experiment, the answer generators are random and contain the whole spectrum of difficulties. As such, results are directly comparable to those using a fixed number from Table 5.1.

The table shows that for a series of questions whose distributions can freely spread in the whole spectrum, constant *c* will be able to identify the correct answer in A per cent of the questions. For example, in a set of binomial (O=2) questions, to acquire the right answer in 95% of the questions (A=95%), the required constant (*c*) is 7 and the expected average number of answers is 18.97 (N_A).

As we can see, with all other things being equal, ANA can consistently provide at least the same accuracy with fewer answers, when compared to the fixed number approach. For the previous example, the fixed number approach would require 53 answers per question, while ANA can achieve the same result in only 18.97 answers (on average), a reduction of ~65%. The phenomenon is consistent throughout the table, while greater A and O exacerbate the result up to a reduction of an order of magnitude. The practical implications of this, is that any crowdsourcing system is better off using this methodology even with a high constant, rather than the fixed number of answers approach.

Table 5.2. Each cell shows the minimum *c* that achieved the shown accuracy, accompanied by the average number of answers that this produces.

Number of Options (O)	2		3		4		5		6	
Accuracy (A)	c	N_A	c	N_A	c	N_A	c	N_A	c	N_A
80%	2	3.12	2	3.85	2	4.22	2	4.39	2	4.45
85%	2	3.12	3	7.43	3	8.20	3	8.49	3	8.57
90%	4	8.77	5	15.80	5	17.33	4	13.00	4	13.11
91%	4	8.77	5	15.80	5	17.33	5	17.84	5	17.96
92%	4	8.77	6	20.39	6	22.35	5	17.84	5	17.96
93%	5	12.00	6	20.39	6	22.35	6	22.95	6	23.08
94%	6	15.41	7	25.20	7	27.58	7	28.30	7	28.45
95%	7	18.97	9	35.39	9	38.65	8	33.85	8	34.00
96%	8	22.66	10	40.68	11	50.34	10	45.49	10	45.68
97%	10	30.34	14	63.04	14	68.75	13	63.91	13	64.14
98%	14	46.60	19	93.06	19	98.04	19	103.56	18	96.94
99%	24	90.43	33	184.05	33	200.03	32	196.50	32	197.01

To better explain how this is possible consider the following. By examining the difference between the two most popular answers we get an estimation of the question's difficulty. This allows stopping easy questions sooner than hard ones. Consequently, the answers not used at the easy questions can be used as extra answers to the hard ones, and thus increase the accuracy through a better distribution of answers, while maintaining the same number of answers.

Similarly to the previous experiment, allowing the questions to range all the way from the easiest to the most difficult, creates a rather unrealistic scenario. Like before, we have repeated the experiment for questions where the first and second most popular options of the distribution have at least 30% difference, and no more than 80% (e.g. for O=2, distributions can range from A=90% B=10% to A=65% B=35%). For O=2, 99% accuracy can be consistently achieved with c=5 and requires no more than 10 answers on average. For comparison, the same number of the previous experiment was 22.

It should be noted that in the 10m repetitions, and especially for the bigger constants, some questions required a very big amount of answers before they could satisfy the condition. The maximum noted was 7352, which of course is completely unrealistic for any crowdsourcing system. It would thus be reasonable that for real applications the maximum number of required answers should be capped to a reasonable estimate. Questions that hit that cap before they can satisfy the condition can be treated similarly to questions of the fixed

number approach, where the most popular result is accepted as the correct answer. In essence this creates a hybrid between the two approaches, where if the question can be stopped sooner it does, but if it does not the default fixed number is used instead. This creates a system where it is impossible to end up requiring more answers, so the only risk is a potentially loss in accuracy from questions that might be erroneously stopped too soon. Another approach would be to defer questions that hit the cap to experts for further examination, as this is an indication that they might be too hard for the crowd to answer.

Hybrid ANA has been further tested with the real data that occurred from the Logic Controller via the Crowdsourcing Platform of ACIBa, and presented along with the rest of the system's evaluation in the following chapter (section 6.1).

5.3 Summary and Conclusions

Crowdsourcing systems are strongly affected by the amount of answers required for each question. The traditional approach of using a fixed number of answers, is potentially using more resources than really required. Not only it is difficult to select an appropriate and justifiable number, but often this number might affect different questions differently with adverse results for the system. The first part of this work suggested an easily reproducible lab experiment that can be used to test the accuracy of a crowdsourcing system. The produced table for fixed number of answers can be used as a baseline for comparison for other, smarter systems. In the second part we have proposed ANA, a methodology that allows changing the number of answers gathered for each individual question. It was experimentally shown that under similar conditions, ANA will always outperform the fixed number of answers.

An important aspect of ANA is its ability to produce results in the absence of information concerning user quality. However, it is equally important that it can be easily adjusted to include this information, with the two approaches being complimentary rather than antagonistic. The adjustment is as simple as calculating the difference between the weighted scores of the first two options, rather than the difference between simple votes. Of course the final condition to be met will need to depend on the user weighting methodology, the main principle however remains unaffected. To that end, different metrics of difficulty or user consensus can be used based on the same principle, that "easy" questions can be stopped sooner than "hard" ones. For example the number of options could be part of the stop condition (e.g. 4+log(NumOfOptions), instead of just 4), or a more holistic metric of difficulty could be used, such as the entropy of the already acquired answers.

6 DEMONSTRATION AND RESULTS

To demonstrate the use of ACIBa a proof-of-concept system was developed for compact digital cameras. The system was 'branded' as MarketTroll and was publically accessible at www.markettroll.co.uk, or through the university's subdomain at markettroll.ee.port.ac.uk. In the back-end MarketTroll is a Java based application with a web interface. All information is read from and saved to files, no database was used. The site remained live until the analysis was completed with the help of the crowd. Over the course of 3 months, 150 workers gave in more than 3000 answers in over a 100 questions. Upon the completion of the analysis, the site was exhibiting the final recommendations for a further 5 months, while an online survey was capturing user's feedback on the system's performance.

This chapter begins by presenting an additional evaluation of the hybrid ANA methodology presented in chapter 5, using the real data gathered. It then continues to present the implementation details and the full operational cycle, beginning with the data gathering from e-shops, continuing with the analysis through the classifiers and crowdsourcing and finishing with scoring and the produced recommendations. This is followed by a presentation of the survey's results, which measures the success of MarketTroll, as well as gauges the potential success of a more fully developed system.

6.1 Testing the hybrid adaptive methodology

The methodology described in section 5.2 was purposefully not applied during the data collection of the Logic Controller. This was done to create a borderline for comparison of the proposed methodology, ANA, against the standard fixed-number approach. In this section we will retrospectively apply ANA to the collected data, and compare its performance against the fixed number of answers that was used. For its crowdsourcing needs MarketTroll has run with a fixed number of 30 answers per question. 67 of the generated questions fit the requirements of ANA (multiple choice, single-answer) and will be used for this evaluation. The questions have between 3 and 5 options. The detailed results of the experiment along with detailed distribution of each question can be found at http://alianos.website/r?c=sste.

This experiment examines what would be the effect of ANA, if it has been used instead of the fixed number of answers. This is achieved by selecting the answers one by one in the real order they came in. However, instead of stopping at 30 answers, the process stops when the adaptive condition is satisfied. As only 30 answers are collected, the process also stops if all 30 answers have been used, resulting in an application of the hybrid ANA methodology. In practice, this means that—as far as the number of answers used is concerned—it is impossible for this test to perform worse than the original approach. As a result the interest shifts to finding out what is the cost in accuracy for this potential reduction of the average

number of answers required. The effects of different constants *c* are examined, and the results are presented in Table 6.1.

Table 6.1. Achieved accuracy for different stop condition constants (*c*)

Constant (*c*)	2	3	4	5	6	7
Average Accuracy (%)	88.06	97.01	97.01	98.05	98.05	100
Average Number of Answers	3.58	7.52	9.77	12.09	12.87	15.95
Caps Reached	0	2	6	9	11	14

For example, with a constant *c*=3, in 65 out of 67 examined questions, ANA would select the same answer, giving an accuracy of 97.01%. The average number of answers required before a difference of 3 is grown, is 7.52. Table 6.1 shows that if ANA had been used instead, with a stop condition of 7 and a hard cap at 30 answers, the exact same results could have been produced while requiring only 15.95 answers on average. It is important to note that the experiment does not examine if the result is correct or not, but rather, if the result is the same with the one that was acquired from the fixed number of answers methodology. As it can been seen, all things being equal, the same result could have been achieved with almost half the answers, with no loss in accuracy. As the constant *c* decreases, both the accuracy and the average number of answers used decreases with it, and as a result *c* acts as a compromise point between the two.

It is important to state that the difficulty levels of the questions are spread in the whole spectrum. Once again, we define difficulty as the difference in percentage between the most popular answer, and the second most popular answer (as these occur from the 30 collected answers). As a result difficulties 1-20 represent the most difficult questions and 80-100 the easiest. Figure 6.1 shows the distribution of difficulties of the examined questions. The fact that difficulties are rather evenly spread, implies that these results represent a rather average case of improvement. If questions were mostly on the easy side, the difference would build up faster with less chances for wrong results. In an opposite scenario where questions tended to be hard, more questions would reach the cap before they were able to build the difference, or in the case of a very small *c* there would be increasingly more opportunities to select the wrong answer. As a result the ANA would exhibit even greater reduction in accuracy for smaller *c*, and at best the same performance with the fixed number approach for larger *c*.

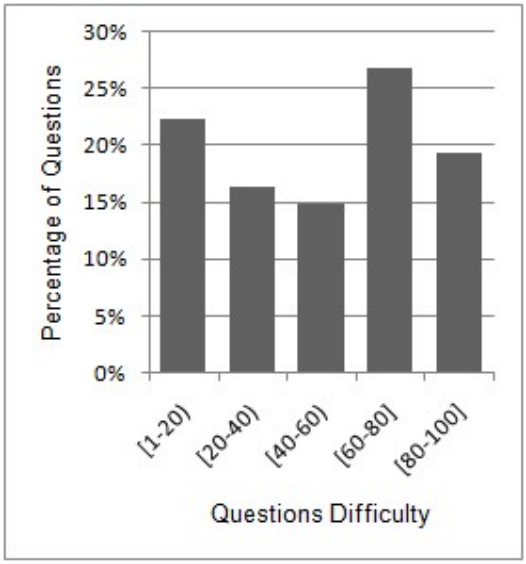


Figure 6.1. Distribution of difficulties of the real questions. Smaller numbers represent harder questions.

6.2 Application of ACIBa

The application of the methodology begins with the data extraction from the web, and continues with the analysis of the product category. Data gathering was performed through custom webpage extractors built for 4 commercial web-sites, namely Argos (190 products), Curries (30), Jessops (116) and Tesco (407). The webpages were extracted using the custom crawler and content extractor described in section 3.2. This data-set is the same with the Camera data-set that was used for the Ensemble Classification evaluation. The dataset was described in detail in Section 4.3.1.

The first task for the crowd is to identify possible usage scenarios of the product, in this case cameras. The generated question prompts workers to either add new usage scenarios or vote on the existing ones. The universally applicable option "Typical everyday use" is automatically seeded, so there is always at least one option that workers can vote for. In addition to that, workers have identified the following two scenarios, "Try to take more artistic photos" and "Shooting often at sports occasions". As a result, the final product of the Category analysis will be the optimal values of important attributes for those 3 identified use cases. The generality of the question allows it to be used with any product category.

The extracted data are used to train the Bullet Classifier (3.4.1), whose outcome is an ordered list of attributes based on their perceived importance. The top 8 attributes from this list are merged with the 8 most commonly encountered attributes in the list of attributes. Attributes that repeat between the two lists are only kept once. The combined list represents the perception of ACIBa on which attributes might be the most important. This information is seeded in the Logic Controller (3.4.4) so that workers can validate which of those attributes

are indeed perceived as important by them. Table 6.2 shows the full list of identified attributes. Column "Reason" describes how each attribute made it to this table, with Bullet meaning it was often found in bullets, and Common meaning it was often found in the attributes' tables. "Times found" shows how often each attribute was found in its respective source.

Table 6.2. Attributes identified as important by ACIBa.

Reason	Attribute	Times found	Score
Common	(lens) optical zoom (x)	195	84.16
Bullet	(video) video recording	36	80.83
Bullet	(general) optical zoom	174	77.50
Bullet	(sensor) resolution	210	75.00
Common	(lens) maximum pixels (mp)	194	70.83
Common	(key information) type	208	65.83
Bullet	(power) battery type	56	65.00
Common	(flash) flash type	189	64.16
Score Borderline: 62.50			
Both	(display & graphics) screen size (in)		50.00
Common	(controls/indicators) macro mode	195	46.66
Bullet	(key information) waterproof	57	44.16
Bullet	(display & graphics) screen type	35	30.83
Common	(features) face recognition	193	25
Bullet	(settings) effects	34	20
Common	(features) smile detection	191	20

Workers were asked to rate the importance of each attribute between 1 and 5. The result of the voting is expressed through a score, where [1-5] awards [0-100] points respectively. The scores are also shown at Table 6.2. Attributes with a score over 62.5 are identified as important and continue to the next step. The limit is arbitrary, and in this case chosen to accept attributes that are just a bit better than the middle (3.5 in the scale [1-5]). It is important to note that selecting a fixed number of top attributes (rather than all attributes over a fixed score) might not be wise at this stage, as the same attribute can appear multiple times with a different name. This was indeed the case for 2 pairs of attributes in the resulting data, nicely illustrated by Table 6.2 (zoom and resolution). If this is the case there is a chance that the reduced attributes will be too few, and that some actually important attributes have been missed.

The next step targets this issue by asking workers to identify similar attributes. A lot of information might be hidden in attributes that did not make it to the original 16 candidates, but in truth are also an expression of one of the attributes identified as important. For example "(lens) optical zoom (x)" and "(general) optical zoom" both made it to the list of important attributes, but "(zoom) optical zoom" did not, even though it expresses the same attribute and as such might contain useful information. For this reason, going through the methodology described in section 3.4.2, LC attempts to match every attribute to every other attribute, but only those pairs that contain at least one identified important attribute are being forwarded to the crowd for validation.

MarketTroll identified 36 possible pairs for merging, from which the crowd validated only 4. Upon inspection, exactly those 4 should have been merged, giving 100% accuracy to the crowd. Table 6.3 below, shows the attributes that were finally merged, while the full list of the considered pairs can be found in Appendix A, Table A.1. This step could be significantly shorter if only merges between two important attributes were considered (as opposed to merges between at least one important attribute and every other attribute). If that path had been followed, workers would have validated 2 out of 2 propositions (instead of 4 out of 36). This is a trade-off between the extra information that the attribute might gain, against the amount of resources required to validate proper pairs. From the 36 propositions the system made, 34 concerned pairs where only one of the attributes was identified as important. From those propositions, only 2 merges are validated by the crowd. As zoom is a lively attribute with a lot of identified values (Table 6.2 above, states it has been found 195 times), the extra information might not have much to offer. As a result, another approach would be to consider merging only important attributes with each other for those attributes where enough information is already known, but with other attributes as well for less populated ones. It is worth mentioning that, in either case, the Attribute Merger correctly identified both pairs of important attributes that should be merged, and in turn the crowd correctly validated both.

Table 6.3. Merged attributes and their respective scores.

Attribute 1	Attribute 2	Score
(lens) optical zoom	(lens) optical zoom (x)	91.11
(general) optical zoom	(lens) optical zoom (x)	86.67
(zoom) optical zoom	(lens) optical zoom (x)	80.00
(sensor) resolution	(lens) maximum pixels (mp)	71.11

As a result of the previous step, from the original 8 attributes validated as important, 4 are merged together in pairs, resulting in a final list of 6 attributes. These 6 form the final list of important attributes, and continue to the next step that is about finding the type of each attribute. Because ACIBa is only able to handle two types of attributes (numeric, and option lists), the crowd is asked to select between these two options, or "Other" (in which case the attribute would be disregarded, but this case did not occur). Table 6.4 shows the result of worker voting.

Table 6.4. Identified types for the important attributes

Attribute Name	Attribute Type
(general) optical zoom	Numeric
(flash) flash type	Option List
(video) video recording	Option List
(sensor) resolution	Numeric
(power) battery type	Option List
(key information) type	Option List

As explained in section 3.4.4, (step 5, p.37) each type follows a different path. Attributes marked as Option Lists go through the process of splitting complicated values to simpler ones, and then reconsolidating those values to as few as possible. The detailed results of this process are presented in Tables A.2 and A.3 respectively, in Appendix A. Numeric attributes have their units automatically merged, and possible values split in ranges. Table 6.5 shows the final list of values for Option List type attributes and the number ranges generated for Numerical type attributes.

At this stage ACIBa has information on which are the important attributes and the possible values these can take. The crowd is finally asked to identify which are the best values for those attributes, for each use case. The final results of this step can also be seen in Table 6.5. Each worker is allowed multiple votes for each question. The options for

'theMoreTheBetter' and 'theLessTheBetter' gather information for the trend of each attribute, but they are not being actively used as part of this research and can be ignored. They are however presented as they were used in the percentage calculations, and omitting them would distort results. The information on this table is passed to the Recommender System, which is going to use it for product scoring purposes.

Table 6.5. The important attributes along with their possible values, and the voting for the best values for each use scenario.

	Typical everyday use		Shooting often at sports occasions		Try to take more artistic photos	
	Votes	%	Votes	%	Votes	%
(video) video recording						
theMoreTheBetter	1	2%	1	3%	4	10%
Y	18	40%	10	26%	10	24%
N	0	0%	1	3%	11	27%
with sound	22	49%	13	33%	11	27%
no sound	0	0%	0	0%	4	10%
Yes 720 HD with sound	4	9%	14	36%	1	2%
totalVotes	45	100%	39	100%	41	100%
(flash) flash type						
theMoreTheBetter	1	2%	2	5%	6	14%
Na	0	0%	0	0%	0	0%
Pop-up	1	2%	6	15%	7	17%
Built-in	23	51%	22	54%	20	48%
Automatic	20	44%	11	27%	9	21%
totalVotes	45	100%	41	100%	42	100%
(key information) type						
theMoreTheBetter	2	4%	5	11%	7	17%
Superzoom	1	2%	8	18%	4	10%
Compact	24	53%	15	33%	13	31%
Digital	18	40%	17	38%	18	43%
totalVotes	45	100%	45	100%	42	100%
(general) optical zoom, unit:"x"						
theMoreTheBetter	1	3%	8	21%	5	12%
theLessTheBetter	0	0%	0	0%	0	0%
less than 5.0	16	46%	9	24%	11	27%
[5.0 - 8.0]	14	40%	10	26%	14	34%
[8.0 - 18.0]	3	9%	7	18%	7	17%

[18.0 - 21.0]	1	3%	4	11%	3	7%
[21.0 - 35.0]	0	0%	0	0%	1	2%
More than 35.0	0	0%	0	0%	0	0%
totalVotes	35	100%	38	100%	41	100%
(power) battery type						
theMoreTheBetter	1	2%	0	0%	0	0%
Lithium ion	9	15%	9	18%	4	10%
NB-5L Li-ion	0	0%	0	0%	0	0%
Ion rechargeable battery	2	3%	5	10%	3	8%
C	0	0%	0	0%	0	0%
BP1130 battery	0	0%	0	0%	0	0%
NB-4L	0	0%	0	0%	1	3%
CR2032	0	0%	0	0%	0	0%
Rechargeable lithium ion	27	45%	25	50%	26	67%
AA	1	2%	1	2%	0	0%
na	0	0%	0	0%	0	0%
AAA	5	8%	2	4%	1	3%
AAA Batteries	3	5%	2	4%	0	0%
Rechargeable EN-EL20	1	2%	1	2%	2	5%
EN-EL20 Li-ion battery	0	0%	0	0%	0	0%
Olympus LI-42B	0	0%	0	0%	0	0%
Inbuilt Rechargeable	9	15%	3	6%	1	3%
LI-70B	0	0%	0	0%	0	0%
Lithium polymer	2	3%	2	4%	1	3%
totalVotes	60	100%	50	100%	39	100%
(sensor) resolution, unit: "MP"						
theMoreTheBetter	0	0%	3	7%	4	8%
theLessTheBetter	1	3%	0	0%	0	0%
less than 14.2	17	43%	5	11%	5	10%
[14.2 - 16.05]	10	25%	8	18%	9	18%
[16.05 - 16.1]	6	15%	8	18%	8	16%
[16.1 - 18.0]	6	15%	14	32%	12	24%
[18.0 - 20.3]	0	0%	4	9%	9	18%
[20.3 - 24.0]	0	0%	2	5%	3	6%
[24.0 - 24.3]	0	0%	0	0%	0	0%
More than 24.4	0	0%	0	0%	0	0%
totalVotes	40	100%	44	100%	50	100%

6.3 Producing recommendations

The database of products evaluated as recommendations is based on the products offered by Currys and Pixmania, in the category of compact digital cameras[12]. The database consists of 228 product listings (114 from each source, coincidentally), which results to 123 unique products due to the repetition between the two sources and multiple colour variations of the same product. Each listing was treated individually, as quite often different colours or sources had different prices, or even different review scores. Because the review scores in both those sites were scarce, review scores were extracted manually from Amazon[13]. If a product was not available from Amazon the review from the respective source was used.

The recommendation engine scores products as described in section 3.5. Each product is finally assigned one score for each use scenario. The score does not rate how good a product is, but how good it is for each use, and as such they serve more as an indication of relevance rather than quality (which is reflected in the rating). For this reason, unless the selection is based on score itself, the recommended products are required to have a score of at least 80% the score of the best scoring product (examples 3-8 below). In larger databases this number could be a lot stricter, as it could also vary in order to better reflect the idea behind the selection criteria.

The following cases were considered as product recommendations.

1. Highest absolute score
2. Highest absolute score, with at least 10 reviews and 4.0 rating
3. Best Price
4. Best Price, with at least 10 reviews and 4.5 rating
5. Most Popular, with at least 4.5 rating
6. Highest rating, with at least 10 reviews
7. Best value for money
8. Best value for money, with at least 10 reviews and 4.5 rating

The general notion of the recommendations is that users will always get recommended a good product, sufficient for their needs, while they can tune their purchase by highlighting a particular aspect. For example users not willing to spend a lot of money can buy the cheapest option available that is still a relatively good product and fitting for the use. Users that prefer to be on the safe side could buy the most popular product, while users that do not care about the cost could buy a mix of the highest specs and ratings regardless of cost.

[12] www.currys.co.uk and www.pixmania.co.uk, last accessed 24/10/2014.

[13] www.amazon.co.uk, last accessed 24/10/2014.

From the examined alternatives, 1, 3, 5 and 8 became the actual recommendations finally served to consumers. The choice was made based on what created good variation of results on this particular instance, but more extensive research should be done to identify what customers are looking for in general. Often one camera became the recommendation from more than one of the above categories and uses. This is a success of the system, since if a camera is highly rated and wildly popular, with good specs and at a low price, then apparently this camera dominates the market and becomes the go-to suggestion for non-educated users; a very solid recommendation. However, for the demonstration purposes of this work and with a limited product database we strove for more variation of results, leaving out seemingly more interesting categories. Appendix B, Table B.1, shows the product selected for each of the above scenario, for each use.

The final recommendations were presented online. An introduction page briefly explained the project and how these recommendations came to be. It then prompted the user to select one of the three identified use-cases, i.e. “typical everyday use”, “Shooting often at sports occasions” and “Try to take more artistic photos”. Each link led to a different page with the specific recommendations (Figure 6.2). Each recommendation is accompanied by the value of the most important characteristics as these were identified from the crowd, as well as the price and review scores. Each product also has an indication explaining why it is being recommended.

Finally, MarketTroll prompted users to take a short survey providing feedback on the quality of results, and the potential use of such an RS.

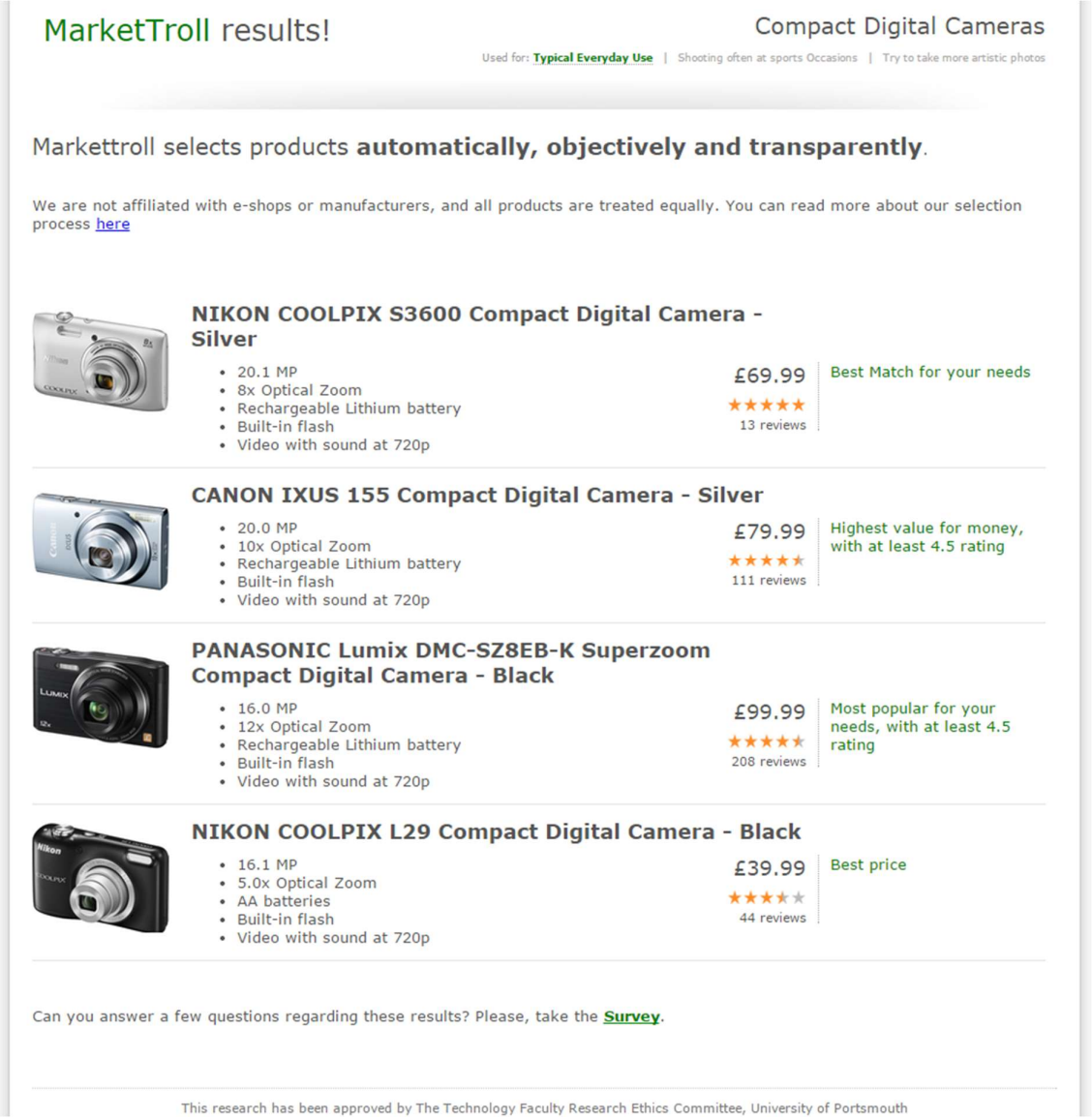


Figure 6.2. Screenshot from the results page served to users of the system, for "Typical everyday use".

6.4 Evaluation of MarketTroll from the users

The ultimate goal of demonstrating ACIBa through a proof-of-concept system was to discover if it can produce a viable recommender system. At this stage, this is mainly reflected in the quality of the results. Consequently, the performed evaluation measures success by that aspect only, and ignores other aspects of a system's success such as usability, acceptability etc. In that aspect, it is also not possible to meaningfully compare the results of MarketTroll with other systems.

As it has been thoroughly discussed, recommender systems simply work towards a different direction, and, to the best of our knowledge, camera-specific guided selling systems only apply a basic filtering to the product database at the end of the guide. For the typical,

non-specialized product, this can yield a few dozens of results. Initially, it might sound reasonable that the two could actually be compared, and it should be the users that decide the amount of results they prefer. However, once the effects of the paradox of choice are considered, it becomes obvious that a comparison between the two would be tainted by the effect, as users have the tendency to prefer the greatest selection regardless of whether this is in their best interest or not. In order to meaningfully compare the two systems, they would have to be used in an actual case scenario where the final effect in sales and the consumer satisfaction could be measured. This is not possible at this stage of development, as in the final sale of a system aspects other than the quality of results play an important role. The lack of resources for a comparison at this scale was also a negatively contributing factor. As a result from the above, to gauge the user's satisfaction and their perceived quality of results a questionnaire was designed and distributed through the web. A sample of the questionnaire can be found in Appendix C.

The questionnaire was put in two different online platforms, namely Typeform and Surviio. A mixture of sampling techniques was used to reach potential participants as widely as possible. The first questionnaire was distributed through social media and snowballing sampling, whereas the second acquired respondents through Cint. Cint is a company that specializes in providing the right audience for sampling, through a large user base and a targeting platform using demographics. The target group for ACIBa (uneducated users not willing to become educated) cannot be targeted with demographics, and as a result no filters were used and the questionnaire was served to the general public. We can assert however that in both cases, respondents were at least users of internet.

Questions where designed to adhere to the design guidelines and principles described by (Brace & Society, 2008) and (Gillham, 2000). The questionnaire examined two factors, the success of the demonstrated system and the potential use of a full scale system. It is formed of 9 questions, 4 questions to address the first factor and 5 questions for the second. The majority were closed questions, whereas three (Q4, Q8 and Q9) were intentionally open ended to allow users to provide their feedback in their own words. The mixed approach (quantitative and qualitative questions) was decided to capture satisfactorily the users' insight and give more informative results as it combined the benefits of two approaches. The reliability of the questionnaire was tested using Cronbach's alpha test for internal consistency. Q1, Q2 and Q3 were eligible for the test and have a Cronbach's alpha index of 0.759 (1st factor). Similarly Q5 and Q6 had an index of 0.926 (2nd factor). Collectively these questions gave an index of 0.851 to the questionnaire. A score over 0.7 is thought of as enough for preliminary research stages, and a score over 0.8 is sufficient for basic research (Lance, Butts, & Michels, 2006). Finally, Chi-square correlation tests were also performed

between Q7, option f (I would not use the system), and Q1, Q2, Q3, Q5, Q6. There was consistent evidence of strong negative correlation. This means that when respondents reply positively in the other questions, they tend to choose option f less in Q7, and vice versa. This is an expected outcome, but because of the opposite nature of the questions it further verifies the internal consistency of the questionnaire.

Initially the questionnaire would take users to the results page previously discussed (Figure 6.2). The users had the opportunity to explore the recommendations, and were then asked to return to the survey to provide feedback. Each user was asked to give feedback only for one of the identified use-case scenarios. The first part measured the extent users thought that the specific implementation of the system succeeded, mainly in terms of results. The second part measured how useful the system was regarded in general, providing feedback on the motivation of this work and gathering information for future work, rather than actually evaluating it.

The survey collected a total of 123 answers; Surviio: 67 (54%), Typeform: 56 (46%). The following section presents the collected results for each question. The majority of questions were analysed quantitatively and presented descriptively, whereas the open ended questions were grouped thematically and presented as tabulated findings. Finally, a Chi square correlation test was performed between the responses of different platforms. The quantitative analysis and correlation test were done with SPSS.

Question 1. If you were to buy a camera, how likely are you to choose one of the recommended options?

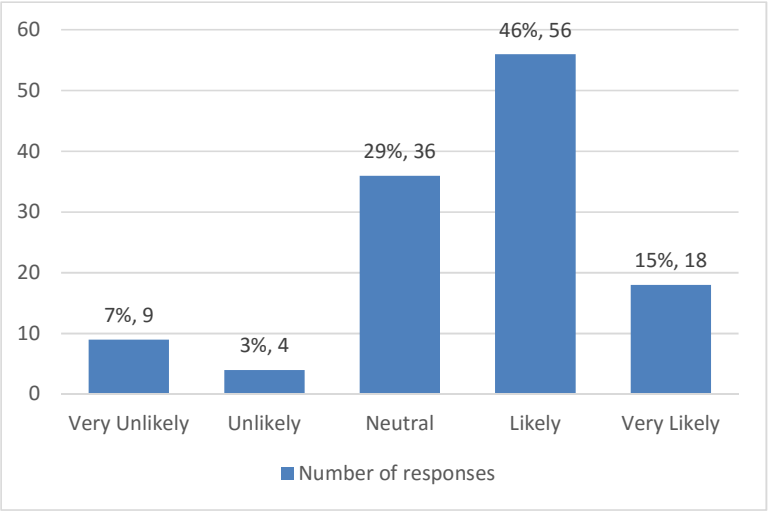


Figure 6.3. Question 1: If you were to buy a camera, how likely are you to choose one of the recommended options?

Question 1 evaluated how likely it is for users to choose from one of the recommended options. This is the main measure of success, as a positive response shows that the produced recommendations are not out of place and constitute valid choices. The cumulative percentage of the respondents giving a rather negative evaluation (very unlikely and unlikely) is 10%, as opposed to a 60% cumulative percentage of the ones that were likely or very likely to choose one of the recommended cameras. Considering that other factors that have not been addressed at this stage might negatively affect the user's choice, as well as the limited original database of products, the least that can be deduced from such a positive response is that the recommended products were good, viable options. In addition, the positive inclination of respondents shows that MarketTroll can, at the very least, assist in decision making, which it turn serves the original goal of reducing information overloading.

Question 2. How do you feel about the variation between the recommended cameras?

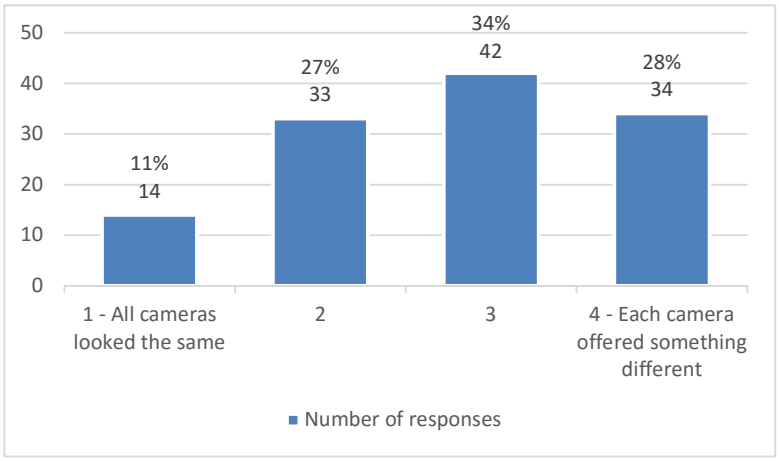


Figure 6.4. Question 2: How do you feel about the variation between the recommended cameras?

Question 2 gauged the user's thoughts on product variation. The question was intentionally designed with an even number of answers to force users into choosing a side. We observe a negative skew in the distribution, with 62% of the respondents feeling that there is good variation between products rather than not. As with Question 1, given the potentially negatively contributing factors these results become even more favourable, demonstrating that in this use case at least, ACIBa was able to produce a recommender system with viable results.

Question 3. In overall, how satisfied are you with our recommendations?

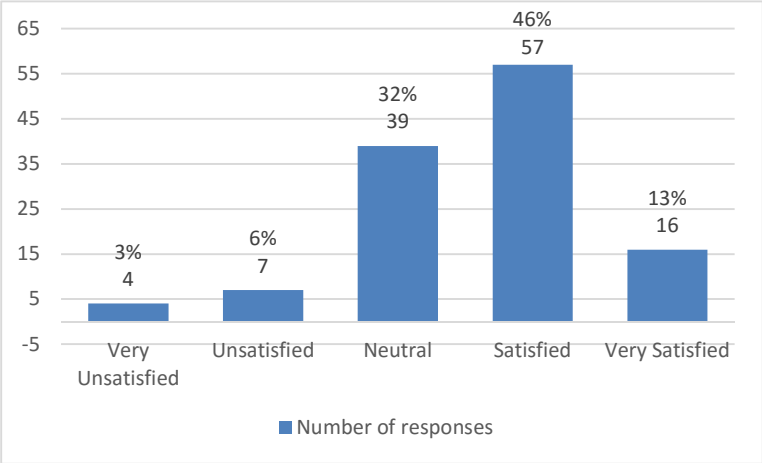


Figure 6.5. Question 3: In overall, how satisfied are you with our recommendations?

Question 3 examined the overall satisfaction of the respondents regarding the quality of the recommendations offered by the system. The question also served the purpose of examining the internal consistency of the questionnaire. Despite a large number of users being undecided (32%), 59% of the respondents were satisfied and very satisfied by the recommendations provided by the system. Results are in line with the previous questions.

Question 4. If you want to justify your answers, tell us what you liked or disliked, or any other comments, please do so here.

The final question for the first factor was open ended to allow capturing of other aspects that users might have found important. The results can be found categorized in Table 6.6.

Table 6.6. Grouping of answers for Question 4

Elements Users Liked	Elements Users Didn't Like	Other Comments
The range of prices, the range of zoom features, and the different looks of each result.	Few Information	The level of importance of the characteristics of the product evaluated is differentiated according what the user is after
The recommendations and criteria. Recommendations nice and compact.	I would like to see more features for each product.	In many cases products are very similar; it's only a matter of finding a good model for a good price.
Good recommendations, with	All recommendations looked	The convenience to store

a good range of prices and a variety of best points for each camera.	the same, except of their color (Similar products recommended)	pictures of the products.
The fact that the user can define the characteristics that finds interesting and get relevant recommendations (the time-saving aspect).	Unclear characteristics, not recommending in simple words or creating comparative tables.	Shooting at sport occasions and taking more artistic photos are mainly the same.
Helpful, Easy to use, Makes choosing easier.	I didn't see anything about the specific modes the cameras can shoot in other special characteristics to make selection easier.	In some cases there is lack of knowledge from the user to understand/interpret the results of the system
A good, simple way of describing the main features of the products and the cost of them.	More specific recommendations based on more specific characteristics at the product and type of scenario (usage).	An explanation of how each feature affects the end result would make the user trust the results more.
Variation in picture analysis, price range, big brands, customer reviews	To be able to read the reviews.	
Good variation of brands, price and features	The recommendations seemed to be at more high price range from what I would be looking at	
The small number of recommendations makes the product selection easy	In many cases the products are too different to be meaningfully compared	
There is variance in specifications and prices		
Very Interesting and Different		

It is clear from the results that the majority of the elements users did not like, as well as their further recommendations, are related to too little information being available. Undoubtedly, too much information would be counterproductive, but as we have seen in the literature enough justification should be provided to increase the user's trust in the system. As the system was not interactive at this stage, users had no guidance neither in how these recommendations came to be, nor in how to best select among the produced options. Both these issues can be addressed by more informative and interactive design of the User Interface (UI), since the required information have been generated in the process.

Such feedback relates more to the UI rather than the results themselves, and is more of assistance to future work rather than evaluation. In overall, this does not seem to have

hindered the perceived quality of the produced results that the majority seems to enjoy, even though results themselves come only from a really small database of products. Explanation or the results however should be addressed in a full scale system.

Question 5. Assuming a final, out-of-the-lab, version of the system. How likely are you to consult it in order to save time?

Question 6. Assuming a final, out-of-the-lab, version of the system. How likely are you to consult it in order to find products better suited for you?

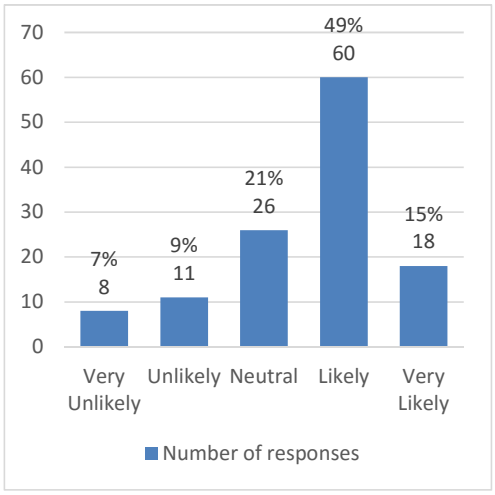


Figure 6.6. Question 5: Assuming a final, out-of-the-lab, version of the system, how likely are you to consult it in order to save time?

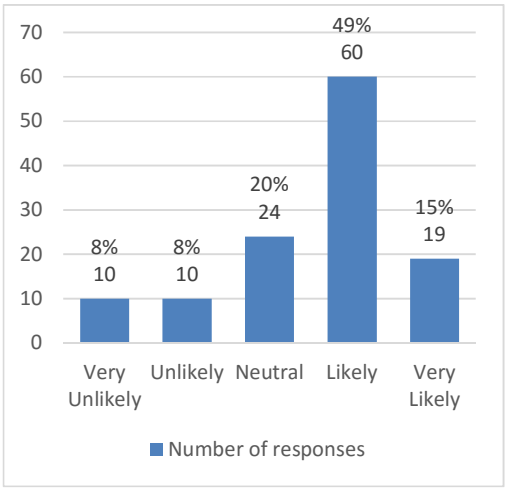


Figure 6.7. Question 6: Assuming a final, out-of-the-lab, version of the system, how likely are you to consult it in order find products better suited for you?

Questions 5 and 6 are very similar as they both assess the likelihood of consulting the system in the future for different purposes. Apart from the obvious destinction between potential uses, they mainly work as a pair to measure the potential use of the system in general, as well as to examine the internal consistency of the questionaire. In both cases roughly 64% reported they are likely or very likely to use the system in the future, ~20% remain neutral and ~16% of the respondents are negative. These results are in line with the findings of the previous questions, and together they show that users have a favorable disposition towards the system, if that would be developed to an acceptable level.

Question 7. In which of the following ways would you be likely to use the system?

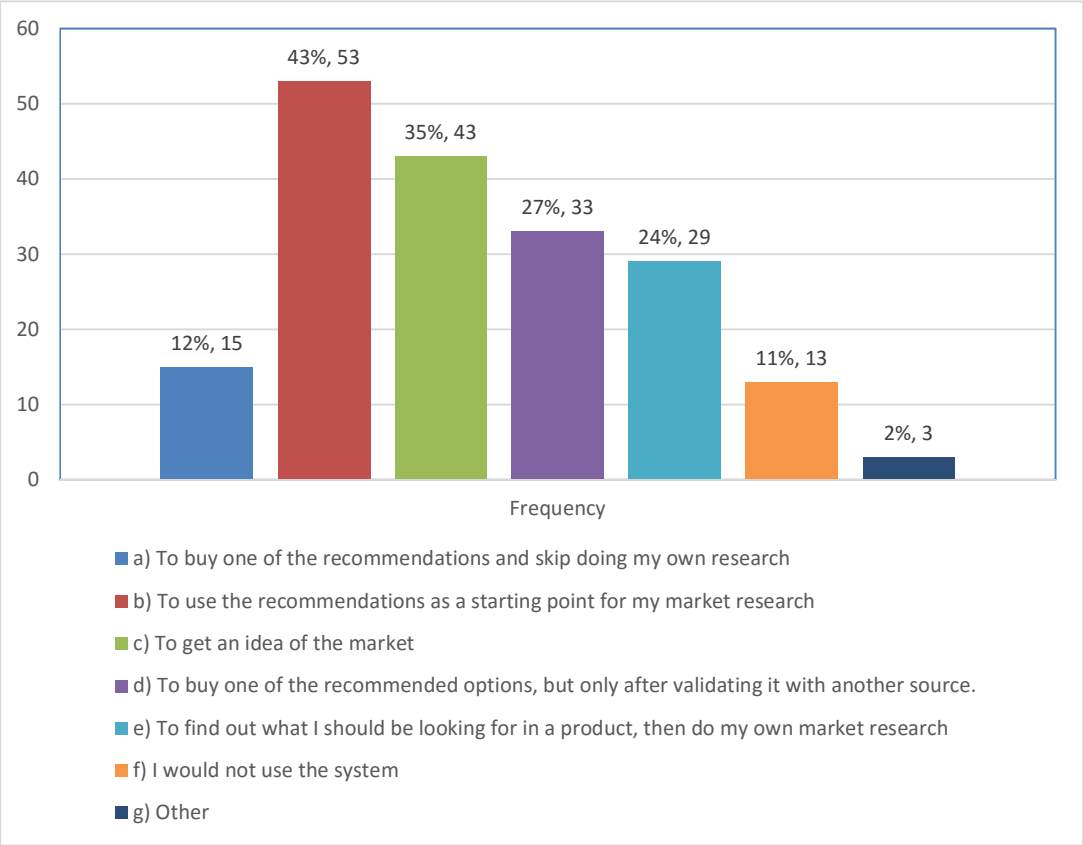


Figure 6.8. Question 7: In which of the following ways would you be likely to use the system? Respondents were allowed multiple answers.

The actual aim of Question 7 was to discover potential applications of an ACIBa based RS. The majority of the responses show that it would be used more as a decision assistance tool rather than a replacement for market research. As it can be seen, only 12% of the users would be willing to just buy one of the recommended products. This is possibly an outcome of the limited user's trust in the system to act with benevolence and competence. 3 respondents chose "other" and gave further explanation on their answer. One of those responses nicely illustrated that point by saying *"I'd only pick one right away if I knew why this and not another".* It would be reasonable to expect that with more explanation about how the choice was made, the alternatives compared, and the why other items have been discarded, these results would skew more and more towards the first option. Nevertheless, the overwhelming majority of respondents stated that they are likely to use the system for one reason or another, compared with 11% who said that they would not use the system at all. The responses are also in the same path as Q5 and Q6.

It should be noted that respondents where allowed multiple answers, and that the options were served in a random order to each respondent, so as to avoid order bias.

Question 8. At the moment, the system is searching for the following products to recommend: Best value for money, Best match, Most popular, Best price. Is there another recommendation that you would like to see?

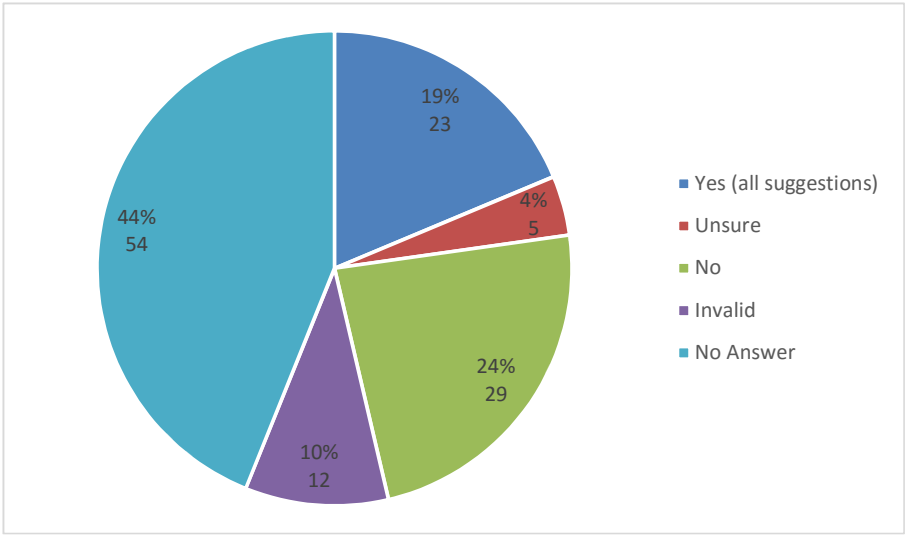


Figure 6.9. Aggregated answers to Question 8.

Question 8 was an optional, open ended question, where users had the chance to provide feedback. It was designed to gauge what aspects of product recommendations are important, mainly gathering information for future research and further iterations of the system. The results have been grouped into five categories. “No” contains all answers that explicitly said no, as opposed to “No answer” where the question was left blank. “Yes” shows the number of users that made suggestions for other metrics, while “Invalid” groups together answers that for any reason are not proper responses to this question. Finally “Unsure” contains responses where the user did not clearly say no, neither made a suggestion, but rather replied with “not sure”, “not certain”, “maybe” etc.

From the 66% of the respondents that chose to reply, the majority found the existing recommendations adequate (No, 24%), though the portion of the users who made recommendations is not far behind (Yes, 19%). “Ease of use” was suggested 6 times and was the most common suggestion, “Highest Rating”, and “Most features” came second, while the remaining suggestions are rather fragmented with only 1 or 2 mentions each, and often include features specific for cameras. The general fragmentation of responses is perhaps indicative that it is not a particular suggestion missing, but rather the lack of more options

per se. Finally, suggested metrics such as “ease of use” and “most features”, are features that have not been evaluated as part of the recommendation algorithm. These features are generic and not related to cameras, and could be added as aspects of the product that need to be estimated during the product analysis in future iterations of the system.

Question 9. If you have any further comments, please leave them below.

The final open question allowed respondents to freely give any other feedback. Only 12 users chose to reply, and mainly raised the issue of further explanation on the recommendations. One user directly mentioned trust, by *saying “My main thought is can I trust this thing?”*. Other comments regard mainly the UI, and are less relevant at this stage of the system’s development.

Examining for bias between sampled groups

Finally, we examined if there are any statistically significant differences in the answers acquired through social media (Typeform questionnaire) and Cint (Surviio questionnaire). This was investigated mainly to find out if the acquisition of answers through social media causes any favourable bias (self-selection). Chi-square test was used to find evidence of possible correlation between the questionnaire platform used and user satisfaction. We note that according to the Monte Carlo significance index; values less than 0.005 show a significant correlation. The summarized results are shown in Table 6.7. Results revealed no evidence of significant correlation, and we can thus assume that both samples have the same bias.

Table 6.7. Chi-square tests between Questionnaire Platform and Q1, Q2, Q3, Q5, Q6 and Q7.

	Q1	Q2	Q3	Q5	Q6	Q7
Monte Carlo Sig (1-sided)	0.277	0.373	0.076	0.094	0.148	a) 0.569 b) 0.409 c) 0.134 d) 0.422 e) 0.056 f) 0.006 g) 0.433

6.5 Summary and Conclusions

MarketTroll was developed as a proof-of-concept system based on ACIBa. The system served various purposes; to test the adaptive number of answers methodology presented at Chapter 5, to showcase the use of ACIBa, as well as to act as a demonstration and evaluation use case.

The application of hybrid ANA on real data produced findings in line with the theoretical conclusions from the previous chapter. Hybrid ANA was shown to outperform the fixed number of answers, since our tests show that the same results could have been achieved with roughly half the answers.

Going forward, MarketTroll was able to complete its operational cycle and produce recommendations. The user satisfaction survey revealed that that these recommendations were at least acceptable, since the majority of the respondents stated that they would be likely to select one of the recommended items. We also find indications that such a system would find use as a decision assistance tool, which in turn can reduce information overloading. The least that we can deduce from that feedback, is that the methodology followed by ACIBa is able to produce rational and viable recommendations, making it a possible new approach to RS. Lack of a more detailed of explanation for the results, as well as the evaluated alternatives, was often highlighted as a limitation of MarketTroll. However, this was likely exacerbated in MarketTroll as it is only developed to the stage of proof-of-concept, while ACIBa does produce much of that information during the data analysis. A fully developed system would have the ability to better present, or otherwise utilise such information.

7 CONCLUSIONS

This thesis has discussed the issue of information overloading in online markets. As it was shown, the existing Recommender Systems techniques do not necessarily assist consumers to decide which product is the best for them, and product filtering requires that consumers know what filters to use. Guided selling might well be the online equivalent to the seller's advice for online markets, but there are many challenges in its development and adoption.

This work has created a novel Recommender System Framework that can restrict the amount of options available to the consumer to a few relevant alternatives. The framework replaces the expert used in traditional guided selling with knowledge extracted from a crowd of potentially non-experts. The ultimate achievement of an RS created within this framework, is to reduce information overloading by offering less choices with meaningful trade-offs.

Apart from the framework itself, ACIBa has made several significant contributions that flow directly from its development. Specifically, this work has made the following contributions:

1. A novel recommendation framework, based on a dynamic combination of Artificial and Crowd Intelligence, aimed at restricting the amount of recommended options.
2. A Crowdsourcing Platform (CP) in the form of reusable, open-source software, which implements a basic crowdsourcing interface.
3. A novel ensemble classification methodology that can utilize training data which are not typically usable in traditional supervised classification.
4. A simple methodology that can reduce the number of answers required by a crowdsourcing system, and can operate in the absence of any other quality criteria.

7.1 Discussion and future work

The main contribution of ACIBa is that it offers a pathway to automated, "objective" and transparent product recommendations of only a few products. The word "objective" is used in quotes because ultimately it is always someone's criteria that creates the recommendations. However, ACIBa utilises a large, arbitrary crowd of people that only collectively create the marking criteria, which in turns minimizes the risk for malicious recommendations and personal views. This is comparable to the change from professional to user reviews, with the additional benefit that because users of ACIBa do not know which questions will be served, the system cannot be as easily manipulated.

At the heart of ACIBa is the Logic Controller (LC), which is in essence a well-defined list of steps that begin with raw data, and finish with a set of criteria upon which recommendations can be formed. This manipulation of data is important for two main reasons. The first reason is that LC is a great example of how crowdsourcing can be used to solve more complicated problems, through segmentation to smaller problems and the combined power of artificial and human intelligence. LC tackles a series of such problems, such as the consolidation of different attributes that refer to the same thing or the separation of values that refer to more than one thing. This happens by a classifier doing most of the work, and the crowd validating and correcting the outcome. Similar techniques could be applied to different problems where the bulk of the work can be automated, but the software is unable to handle difficult or abnormal cases, or it cannot be trusted to take decisions and as such results need to be validated. With the demonstration of LC, it was shown that crowdsourcing can be used to compliment the operation of classifiers, rather than merely help train them.

Finally, the outcome of LC is usable information in itself, since even when detached from a product recommender, it can offer guidance to the uneducated consumer as to what to look for and consequently what filters to apply.

The Crowdsourcing Platform (CP) proved to be a very effective way of communicating with the crowd. The generality and extensibility of the platform allows it to accept and handle any question, which in turned provided an easy way to overcome obstacles in data generation. CP can be potentially reused without a crowd, merely as an interface between humans and software, offering to the latter access to human intelligence. A single (or a few) experts can substitute the crowd, which in essence converts CP to a dynamic data input interface. Of course the platform is far from perfect or complete, but still, it has proven more than adequate during its use in the scope of this work. Its availability as open-source software and built-in extensibility should allow any developer to customize the system to their needs.

The Ensemble Classifier discussed in Chapter 4, contributes to the field of ensemble classification by proposing a methodology that can utilize training data that are not in the same format with the texts that need to be classified. This opens up the possibility to tap into many data sources that were previously unusable. The proposed methodology differs dramatically from other ensemble classification approaches such as bagging and boosting. It uses classifiers that are inherently different from each other, rather than diversified instances of one base classifier. The methodology can be generalized and transferred to other domains that can utilize bespoke solutions, so long as a database of information can be used as training data. By allowing the use of training data that would be otherwise unusable, it can

save the considerable amount of time and effort that is typically needed to hand-tag training instances.

At its current state the process is not fully automated as the weights of each member classifier need to be decided by the developer. Even though it does not take any more training instances to tune the weights than those required to test and develop the member classifiers, the design can benefit greatly from a mechanism that can automatically assign them. Pursuing this direction, it would be interesting to identify ways that can adjust the weights between classifiers per instance, rather than using the same fixed weights for every classification. This is an excellent lead for further research, with a promising starting point being the use of the confidence levels of each classifier as a weight, or as a weight modifier.

On the way, light has been shed on what constitutes a proper feature set for classifiers specifically tailored for consumer electronics. In addition, the presented empirical evaluation can be used as a starting point for future researches, by providing the initial parameters.

The proposed Adaptive Number of Answers (ANA) methodology and its hybrid variation, have proven to greatly outperform the traditional fixed number of answers. The baseline of fixed number of answers has been already improved by other researchers, through methods that typically use an estimation of the user's quality or some other quality metric of the answer. The ability to work in lack of such data is one of the major advantages of ANA, but still a comparison to such methodologies could prove beneficial. The same could be true for the combination of the two in the search for an even more efficient methodology. Finally, the code used to benchmark ANA is also made publically available, and could be reused by other researchers to benchmark their work.

Even though not explicitly proven for ACIBa, all relevant literature suggests that reducing the amount of alternatives is a solution to information overloading. Unlike other recommender systems (e.g. collaborative filtering), the limitation of products to just a few is not arbitrary, such as the top 5 better scoring products, but is based on specific criteria that are understandable and comparable by the consumer. This is of outmost importance since in a different case the user will still feel the need to examine the rest of the market for alternatives, defeating the original purpose of the system. This is also the reason why trust and transparency are highlighted in the literature as important challenges in the adoption of new recommender systems. As the processes followed by ACIBa are traceable and transparent, it can be demonstrated to the user which criteria are being used, how they came to be, and how every other product has scored against those criteria. Furthermore, the criteria can be adjusted based on factors the user understands, such as price, quality, popularity etc.

This is reflected in the user evaluation of MarketTroll that acted as a use case of ACIBa. Despite the limited database, its infant stage and the lack of time to build the users' trust, users have clearly looked favourably at the system. The lack of explanation in the interface has been highlighted, but on the other hand so was simplicity. One of the key takeaways of the evaluation is even though users would be reluctant to trust the system and purchase one of the recommendations right away, they found them plausible decisions, and would be willing to purchase one of the recommendations after validating with another source. Even if MarketTroll was not able to completely abolish product overloading, it was able to offer meaningful assistance in decision making and reduce the phenomenon.

In conclusion, this work has met both its objectives; creating an RSF and demonstrating the relevant RS. The RSF offers two major key components, a classifier for the work that can be done with artificial intelligence, and a crowdsourcing platform for the work that requires human intelligence. The combination of those two creates a novel framework within which complicated problems can be solved. This work has made further contributions to the respective areas of its main components. The proof-of-concept system managed to complete a full operational cycle, from data-gathering to product recommendations. Despite its infant state, there are clear indications that MarketTroll was able to reduce the phenomenon of information overloading.

REFERENCES

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 19–26). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1148177

Andrews, P. (2005). *Guided Selling: On demand information for buyers*. IBM Advanced Business Institute. Retrieved from http://www-935.ibm.com/services/us/imc/pdf/gt510-6205-guided-selling.pdf

Bachrach, Y., Graepel, T., Minka, T., & Guiver, J. (2012). How To Grade a Test Without Knowing the Answers—A Bayesian Graphical Model for Adaptive Crowdsourcing and Aptitude Testing. *arXiv Preprint arXiv:1206.6386*. Retrieved from http://arxiv.org/abs/1206.6386

Barr, J., & Cabrera, L. F. (2006). AI Gets a Brain. *Queue*, *4*(4), 24–29. http://doi.org/10.1145/1142055.1142067

Benbasat, I., & Wang, W. (2005). Trust in and adoption of online recommendation agents. *Journal of the Association for Information Systems*, *6*(3), 4.

Brace, I., & Society, M. R. (2008). *Questionnaire design: how to plan, structure and write survey material for effective market research* (2nd ed.). London: Kogan Page. Retrieved from http://capitadiscovery.co.uk/port/items/874862

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Burke, R. (2007). Hybrid web recommender systems. In *The adaptive web* (pp. 377–408). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-72079-9_12

Burke, R., & others. (1999). Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce* (pp. 69–72). Retrieved from http://www.aaai.org/Papers/Workshops/1999/WS-99-01/WS99-01-011.pdf

Castano, S., Ferrara, A., Genta, L., & Montanelli, S. (2016). Combining crowd consensus and user trustworthiness for managing collective tasks. *Future Generation Computer Systems*, *54*, 378–388. http://doi.org/10.1016/j.future.2015.04.014

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In *Machine learning—EWSL-91* (pp. 164–178). Springer.

Dasu, T., Johnson, T., Muthukrishnan, S., & Shkapenyuk, V. (2002). Mining database structure; or, how to build a data quality browser. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (pp. 240–251). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=564719

Degeratu, A. M., Rangaswamy, A., & Wu, J. (2000). Consumer choice behavior in online and traditional supermarkets: The effects of brand name, price, and other search attributes. *International Journal of Research in Marketing*, *17*(1), 55–78.

Dhar, R. (1997). Consumer preference for a no-choice option. *Journal of Consumer Research*, *24*(2), 215–231.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple classifier systems* (pp. 1–15). Springer. Retrieved from http://link.springer.com/chapter/10.1007/3-540-45014-9_1

Eppler, M. J., & Mengis, J. (2004). The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines. *The Information Society*, *20*(5), 325–344.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, *3*, 1289–1305.

Franklin, M. J., Kossmann, D., Kraska, T., Ramesh, S., & Xin, R. (2011). CrowdDB: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (pp. 61–72). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1989331

Freund, Y., Schapire, R. E., & others. (1996). Experiments with a new boosting algorithm. In *ICML* (Vol. 96, pp. 148–156). Retrieved from http://www.public.asu.edu/~jye02/CLASSES/Fall-2005/PAPERS/boosting-icml.pdf

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*(2), 131–163.

Ghani, R., Probst, K., Liu, Y., Krema, M., & Fano, A. (2006). Text Mining for Product Attribute Extraction. *SIGKDD Explor. Newsl.*, *8*(1), 41–48. http://doi.org/10.1145/1147234.1147241

Gillham, W. E. C. (2000). *Developing a questionnaire*. London: Continuum. Retrieved from http://capitadiscovery.co.uk/port/items/641708

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., & Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI* (pp. 439–446). Retrieved from http://www.aaai.org/Papers/AAAI/1999/AAAI99-063.pdf

Guided Selling. (2005, May 1). Retrieved 15 May 2015, from http://risnews.edgl.com/old-magazine/Guided-Selling39612

Guy, I., Jaimes, A., Agulló, P., Moore, P., Nandy, P., Nastar, C., & Schinzel, H. (2010). Will recommenders kill search?: recommender systems-an industry perspective. In *Proceedings of the fourth ACM conference on Recommender systems* (pp. 7–12). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1864713

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Häubl, G., & Trifts, V. (2000). Consumer decision making in online shopping environments: The effects of interactive decision aids. *Marketing Science*, *19*(1), 4–21.

Hawley, M. (2012, November 19). Best Practices for Online Guided-Selling. Retrieved 15 May 2015, from http://www.uxmatters.com/mt/archives/2012/11/best-practices-for-online-guided-selling-experiences.php

Howe, J. (2006a, June). The Rise of Crowdsourcing. *Wired*, (14.06). Retrieved from http://archive.wired.com/wired/archive/14.06/crowds.html

Howe, J. (2006b, June 2). Crowdsourcing: A Definition. Retrieved from http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html

Ipeirotis, P. G., Provost, F., & Wang, J. (2010). Quality Management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation* (pp. 64–67). New York, NY, USA: ACM. http://doi.org/10.1145/1837885.1837906

Iyengar, S. S., Huberman, G., & Jiang, W. (2004). How much choice is too much? Contributions to 401 (k) retirement plans. *Pension Design and Structure: New Lessons from Behavioral Finance*, 83–96.

Iyengar, S. S., & Lepper, M. R. (2000). When choice is demotivating: Can one desire too much of a good thing? *Journal of Personality and Social Psychology*, *79*(6), 995.

Jiménez, D. (1998). Dynamically weighted ensemble neural networks for classification. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on* (Vol. 1, pp. 753–756). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=682375

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 133–142). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=775067

Katragadda, L. (2008, June 23). Making your mark on the world. Retrieved from http://google-latlong.blogspot.com/2008/06/making-your-mark-on-world.html

Kazienko, P., & Adamski, M. (2007). AdROSA—Adaptive personalization of web advertising. *Information Sciences*, *177*(11), 2269–2295.

Kerber, R. (1992). Chimerge: Discretization of numeric attributes. In *Proceedings of the tenth national conference on Artificial intelligence* (pp. 123–128). Aaai Press.

Kittur, A., Chi, E. H., & Suh, B. (2008). Crowdsourcing user studies with Mechanical Turk. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 453–456). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1357127

Kotsiantis, S., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Frontiers in Artificial Intelligence and Applications*, *160*, 3.

Lance, C. E., Butts, M. M., & Michels, L. C. (2006). The Sources of Four Commonly Reported Cutoff Criteria What Did They Really Say? *Organizational Research Methods*, *9*(2), 202–220. http://doi.org/10.1177/1094428105284919

Lee, B.-K., & Lee, W.-N. (2004). The effect of information overload on consumer choice quality in an on-line environment. *Psychology and Marketing*, *21*(3), 159–183.

Lintott, C. J., Schawinski, K., Slosar, A., Land, K., Bamford, S., Thomas, D., … others. (2008). Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, *389*(3), 1179–1189.

Liu, H., & Motoda, H. (1998). *Feature selection for knowledge discovery and data mining*. Springer.

Liu, X., Lu, M., Ooi, B. C., Shen, Y., Wu, S., & Zhang, M. (2012). CDAS: A Crowdsourcing Data Analytics System. *Proc. VLDB Endow.*, *5*(10), 1040–1051. http://doi.org/10.14778/2336664.2336676

Lohse, G. L., & Spiller, P. (1998). Electronic shopping. *Communications of the ACM*, *41*(7), 81–87.

Lohse, G. L., & Spiller, P. (1999). Internet retail store design: how the user interface influences traffic and sales. *Journal of Computer-Mediated Communication*, *5*(2), 0–0.

Maclin, R., & Opitz, D. (2011). Popular ensemble methods: An empirical study. *arXiv Preprint arXiv:1106.0257*. Retrieved from http://arxiv.org/abs/1106.0257

Maclin, R., Shavlik, J. W., & others. (1995). Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In *Proceedings of the 14th international joint conference on Artificial Intelligence* (pp. 524–531). Canada: IJCAI. Retrieved from

http://pdf.aminer.org/000/389/570/combining_the_predictions_of_multiple_classifiers_using_competitive_learning_to.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (Vol. 1). Cambridge University Press Cambridge.

McCallum, A., Nigam, K., & others. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, pp. 41–48). Citeseer.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81.

Mobasher, B., Cooley, R., & Srivastava, J. (2000). Automatic personalization based on web usage mining. *Communications of the ACM*, *43*(8), 142–151.

Mottola, G. R., & Utkus, S. P. (2003). Can there be too much choice in a retirement savings plan. *Vanguard Center for Retirement Research*.

Natrella, M. (2010). NIST/SEMATECH e-handbook of statistical methods.

Naumann, F., Ho, C. T., Tian, X., Haas, L., & Megiddo, N. (2002). Attribute classification using feature analysis. In *Proceedings of the International Conference on Data Engineering* (pp. 271–271). San Jose: IEEE Computer Society Press; 1998.

Opitz, D. W., & Shavlik, J. W. (1996). Actively searching for an effective neural network ensemble. *Connection Science*, *8*(3–4), 337–354.

Perrone, M. P., & Cooper, L. N. (1992). *When networks disagree: Ensemble methods for hybrid neural networks*. DTIC Document. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA260045

Punj, G. N., & Moore, R. (2007). Smart versus knowledgeable online recommendation agents. *Journal of Interactive Marketing*, *21*(4), 46–60.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., & Riedl, J. (2002). Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces* (pp. 127–134). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=502737

Rennie, J. D. (2001). *Improving multi-class text classification with naive Bayes*. Massachusetts Institute of Technology.

Reutskaja, E., & Hogarth, R. M. (2009). Satisfaction in choice as a function of the number of alternatives: When 'goods satiate'. *Psychology & Marketing*, *26*(3), 197–203.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender systems handbook* (Vol. 1). Springer. Retrieved from http://link.springer.com/content/pdf/10.1007/978-0-387-85820-3.pdf

Rogova, G. (1994). Combining the results of several neural network classifiers. *Neural Networks*, *7*(5), 777–781.

Rokach, L., & Maimon, O. Z. (2008). *Data mining with decision trees: theroy and applications* (Vol. 69). World Scientific Publishing Company Incorporated.

Ruck, D. W., Rogers, S. K., & Kabrisky, M. (1990). Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, *2*(2), 40–48.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on*

World Wide Web (pp. 285–295). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=372071

Scammon, D. L. (1977). ' Information load' and consumers. *Journal of Consumer Research*, 148–155.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. In *Applications of Data Mining to Electronic Commerce* (pp. 115–153). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-1-4615-1627-9_6

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.

Schick, A. G., Gordon, L. A., & Haka, S. (1990). Information overload: A temporal approach. *Accounting, Organizations and Society*, *15*(3), 199–220.

Schwartz, B. (2005). *The paradox of choice: Why more is less*. Harper Perennial.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, *34*(1), 1–47.

Senecal, S., & Nantel, J. (2004). The influence of online product recommendations on consumers' online choices. *Journal of Retailing*, *80*(2), 159–169.

Sheng, V. S., Provost, F., & Ipeirotis, P. G. (2008). Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 614–622). New York, NY, USA: ACM. http://doi.org/10.1145/1401890.1401965

Sundaresan, N. (2011). Recommender systems at the long tail. In *Proceedings of the fifth ACM conference on Recommender systems* (pp. 1–6). New York, NY, USA: ACM. http://doi.org/10.1145/2043932.2043934

Tangermann, O., & Streit, J. (2011). *How online shops and brand websites engage with their customers through product recommendations and achieve higher sales*. excentos GmbH. Retrieved from http://www.excentos.com/images/stories/technologie/whitepaper/excentos-whitepaper-guided-selling_en.pdf

U.S. Census Bureau. (2012). *E-Stats* (p. 8).

Van Setten, M. (2005). Supporting people in finding information: hybrid recommender systems and goal-based structuring. Retrieved from http://eprints.eemcs.utwente.nl/9166/

Xhemali, D., Hinde, C. J., & Stone, R. G. (2009). Naïve Bayes vs. Decision Trees vs. Neural Networks in the classification of training web pages. *International Journal of Computer Science Issues*.

Zhang, G. P. (2000). Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *30*(4), 451–462.

APPENDIX A – INTERMEDIATE LC RESULTS

Table A1. Attributes identified as similar enough to be forwarded to the crowd for validation.

Attribute 1	Attribute 2	System Score	User Score
(lens) optical zoom	(lens) optical zoom (x)	0.62	91.11
(general) optical zoom	(lens) optical zoom (x)	0.66	86.67
(zoom) optical zoom	(lens) optical zoom (x)	0.62	80.00
(sensor) resolution	(lens) maximum pixels (mp)	*	71.11
Worker Score borderline: 65.00			
(general information) batteries required	(power) battery type	0.52	57.78
(general) li-ion battery	(power) battery type	*	53.33
(power) battery type	(general) li-ion battery	*	53.33
(video) video recording	(video) hd recording	0.84	48.89
(lens) optical zoom (x)	(general) optical zoom banding	*	46.67
(video) video recording	(general) hd video	0.47	41.11
(general) batteries required	(power) battery type	0.30	40.00
(general) hs system	(sensor) resolution	*	35.56
(lens) digital zoom	(general) optical zoom	0.76	24.44
(video) video recording	(general) video out	0.33	22.22
(general) battery compatibility fitting	(power) battery type	*	22.22
(flash) flash guide number	(flash) flash type	0.62	15.56
(power) battery type	(general) battery compatibility fitting	*	15.56
(flash) flash type	(flash) flash guide number	0.62	13.33
(flash) guide number / flash range	(flash) flash type	0.47	13.33
(flash) flash type	(flash) flash guide number	*	13.33
(flash) flash type	(flash) guide number / flash range	0.62	11.11
(flash) flash type	(flash) flash mode - auto	0.62	8.89
(power) battery life	(power) battery type	0.40	8.89
(flash) flash type	(flash) flash mode - auto	*	8.89
(lens) maximum pixels (mp)	(features) maximum iso rating	0.42	7.78
(power) battery type	(other) battery life	*	7.78
(general information) batteries required	(general) optical zoom	0.73	6.67

(sensor) resolution	(lens) size of sensor (mm)	*	6.67
(sensor) resolution	(lens) sensor type	*	5.56
(viewfinder) type	(flash) flash type	*	4.44
(power) shots per battery charge	(power) battery type	*	3.33
(power) battery type	(power) battery life	*	3.33
(general) batteries required	(general) optical zoom	0.66	2.22
(other) battery life	(power) battery type	*	2.22
(key information) type	(general information) size	*	2.22

Column "System score" shows the calculated similarity score given by ACIBa. Due to a server restart some information was lost. However, it is still known that the missing scores were at least 0.30, since that was the lowest score to be considered for inspection by the crowd. The affected scores are marked with a star (*). Column "Worker Score" shows the score assigned by worker votes, expressed in the scale [0-100]. Scores over 65.00 result in the attributes being actually merged.

Table A2. Values of attributes that were proposed for splitting to multiple simpler values. The result of the split should be two or more values that are acceptable as individual values. The winning votes are highlighted in bold.

Attribute	Value	Symbol	Result		
			Yes	No	Partial
(flash) flash type	Built-in	-	6	**24**	-
(flash) flash type	Pop-up	-	3	**27**	-
(video) video recording	Yes - no sound	-	**23**	7	-
(video) video recording	Yes - with sound	-	**26**	4	
(power) battery type	NB-5L Li-ion	-	5	**21**	4
(power) battery type	Lithium-Ion rechargeable battery	-	**19**	11	-
(power) battery type	EN-EL20 Li-ion battery	-	2	**22**	6
(power) battery type	Rechargeable EN-EL20	-	9	**21**	-
(power) battery type	Olympus LI-42B	-	5	**25**	-
(power) battery type	NB-4L	-	5	**25**	-
(power) battery type	Li-Ion	-	10	**20**	-
(power) battery type	Inbuilt Rechargeable,CR2032	,	**26**	4	-
(power) battery type	LI-70B	-	8	**22**	-

A partial split can occur when the attribute can be split, but not in every occurrence of the symbol. The winning values are highlighted in bold. If a value can be partially split, all parts are submitted as separate values. No such cases occurred.

Table A3. Values proposed for merging, either by the heuristic system procedures, or by recommendation of the crowd.

Source	Attribute	Value 1	Value 2	Result		
				Yes	No	Maybe
Crowd	(flash) flash type	Built in	Built-in	Merged		
Crowd	(flash) flash type	Automatic	Pop-up	Not Merged		
Crowd	(flash) flash type	Automatic	Built-in	Not Merged		
System	(flash) flash type	None	na	**16**	4	10
System	(flash) flash type	na	Built in	Not Merged		
System	(flash) flash type	Automatic	Built in	Not Merged		
Crowd	(power) battery type	Li-Ion	Lithium ion	Merged		
System	(power) battery type	Lithium	Lithium ion	Merged		
System	(power) battery type	Lithium polymer	Lithium ion	Not Merged		
System	(power) battery type	na	Lithium ion	Not Merged		
System	(power) battery type	Rechargeable lithium ion	Lithium ion	Not Merged		
Crowd	(video) video recording	No	N	Merged		
Crowd	(video) video recording	Yes	Y	Merged		
Crowd	(video) video recording	Y	with sound	Not Merged		
System	(video) video recording	no sound	with sound	0	**30**	0
System	(video) video recording	No	Yes	0	**29**	1
System	(video) video recording	Yes	N	Not Merged		

Due to a system restart most of these questions where left incomplete. As there was a lack of resources, it was decided to disable them rather than restart them, as they were very repetitive, easy, and of little value. For those questions only the result is shown. Most questions had over 15 answers when they were disabled.

APPENDIX B – PRODUCT RECOMMENDATIONS

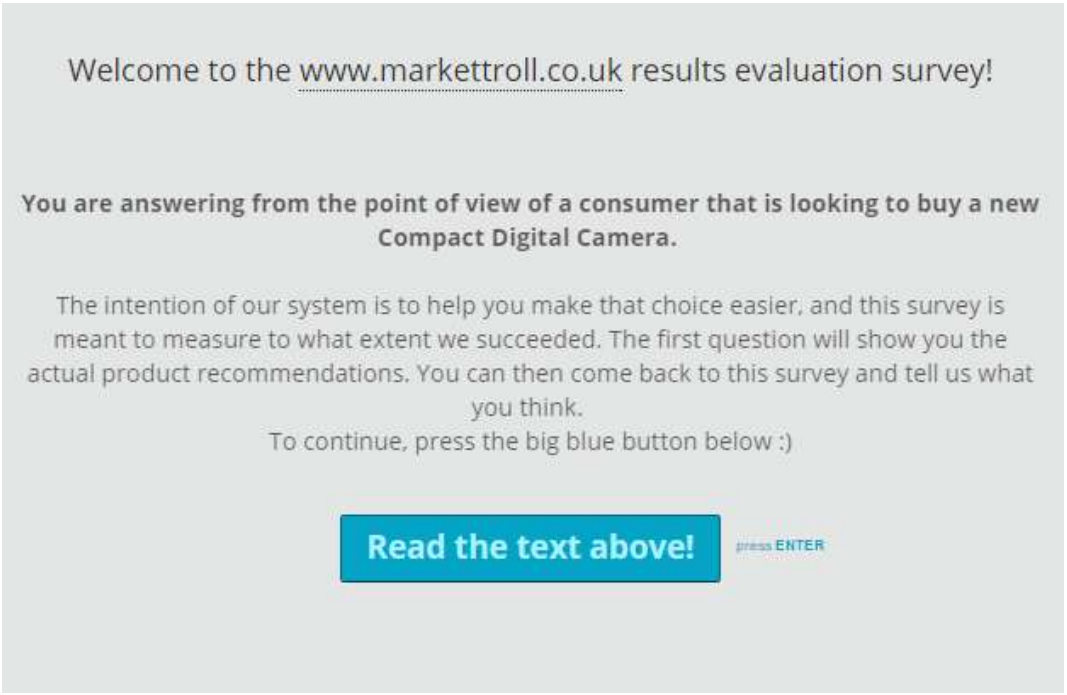
Table B1. Top scoring products for each set of criteria, for each use. This table summarizes the products selected by each set of criteria (out of 123 unique products). Column "Rank" shows where the item has been ranked by the scoring mechanism. Naturally "Highest absolute score" picks the best scoring product. All other products are selected among those that scored no less than 80% of the best scoring product.

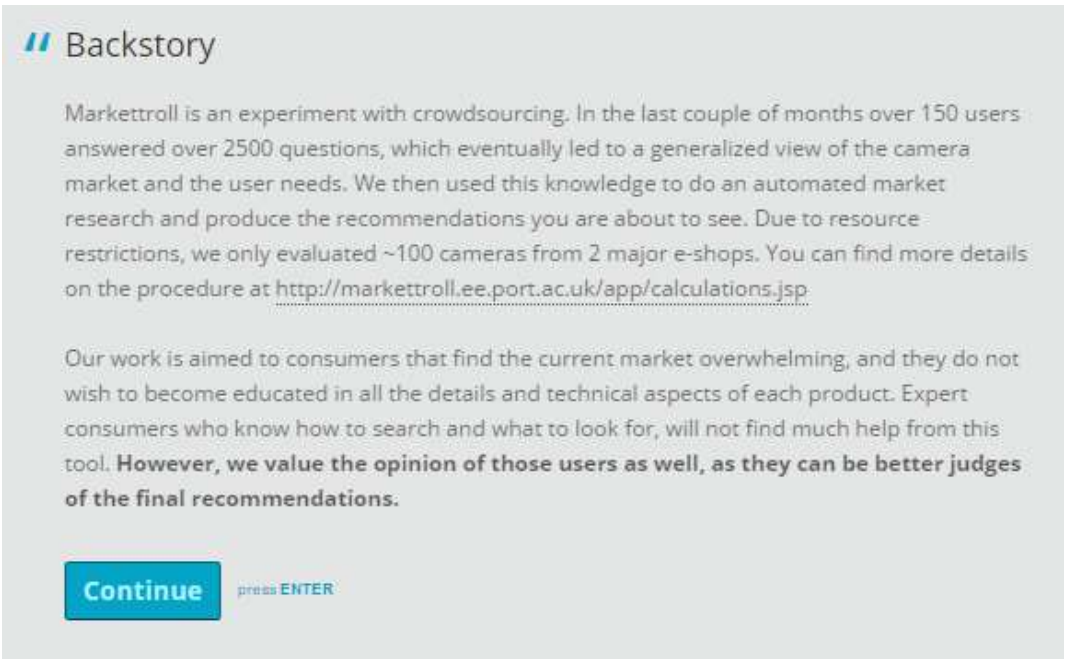
Typical Everyday Use				
Criteria	Product	Rating (Reviews)	Price	Rank
Highest absolute score	NIKON COOLPIX S3600 Compact Digital Camera - Red	n/a (0)	69.99	1
Highest absolute score with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Best price	NIKON COOLPIX L29 Compact Digital Camera - Black	3.8 (44)	39.99	24
Best price with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Most popular	PANASONIC Lumix DMC-TZ10 black	4.4 (390)	235.00	9
Most popular, with at least 4.5 rating	PANASONIC Lumix DMC-SZ8EB-K Superzoom Compact Digital Camera - Black	4.5 (208)	99.99	14
Highest rating, with at least 10 reviews	CANON PowerShot SX600 HS Superzoom Compact Digital Camera - Black	4.7 (28)	119.99	17
Best value for money with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Shooting often at sports events				
Criteria	Product	Rating (Reviews)	Price	Rank
Highest absolute score	NIKON COOLPIX S3600 Compact Digital Camera - Red	n/a (0)	69.99	1
Highest absolute score with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Best price	SONY Cyber-shot DSCW810B Compact Digital Camera	n/a (0)	69.99	4
Best price with at least	CANON IXUS 155 Compact	4.5 (111)	79.99	2

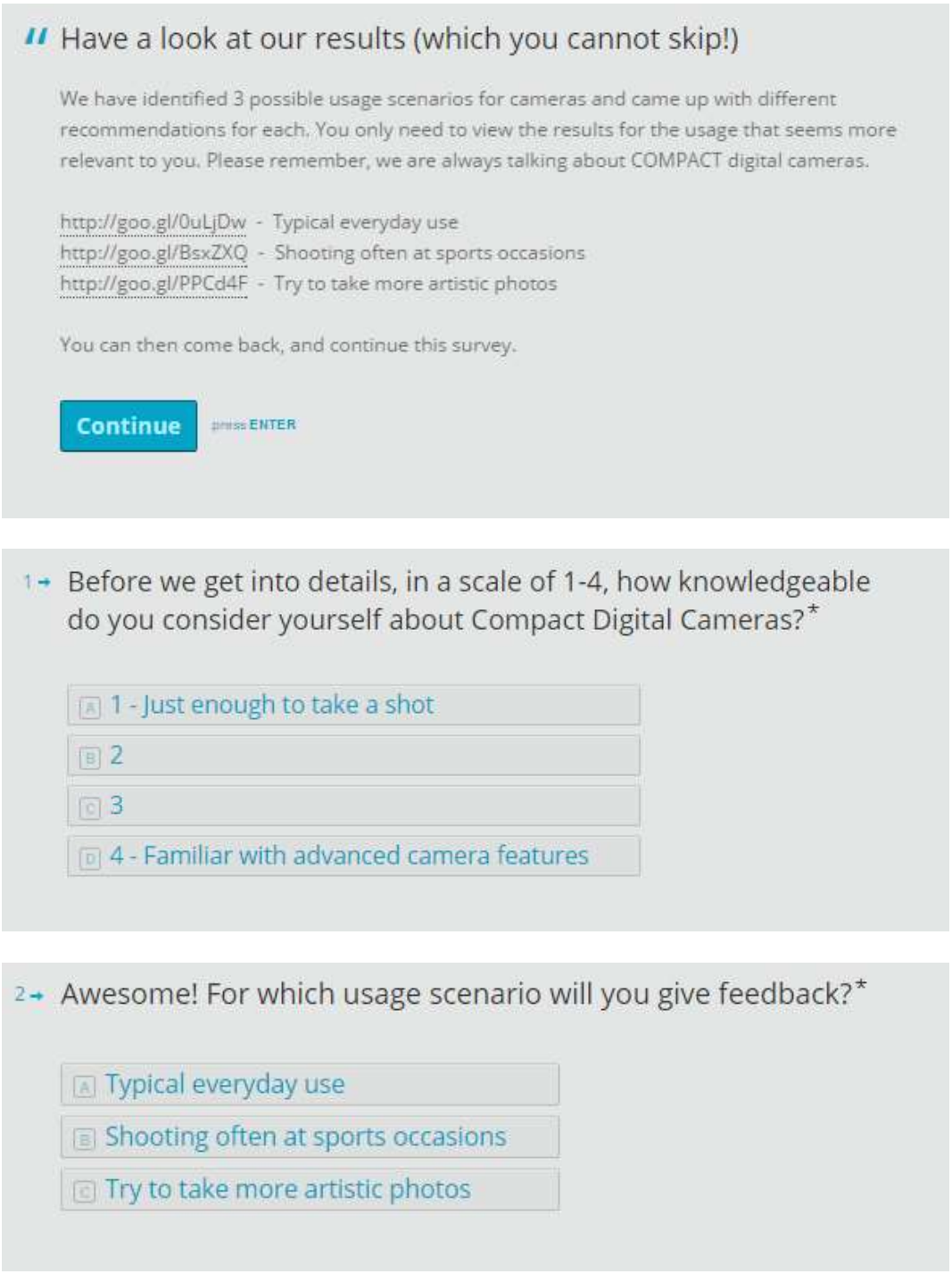
10 reviews and 4.5 rating	Digital Camera - Silver			
Most popular	SONY DSC-HX50B Superzoom Compact Digital Camera	4.5 (136)	179.99	11
Most Popular, with at least 4.5 rating	SONY DSC-HX50B Superzoom Compact Digital Camera	4.5 (136)	179.99	11
Highest rating, with at least 10 reviews	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Best value for money with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Shooting often at sports events				
Criteria	Product	Rating (Reviews)	Price	Rank
Highest absolute score	NIKON COOLPIX S3600 Compact Digital Camera - Red	n/a (0)	69.99	1
Highest absolute score with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Best price	SONY Cyber-shot DSCW810B Compact Digital Camera	n/a (0)	69.99	4
Best price with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Most popular	PANASONIC Lumix DMC-TZ10 black	4.4 (390)	235.00	12
Most Popular, with at least 4.5 rating	SONY DSC-HX50B Superzoom Compact Digital Camera	4.5 (136)	179.99	10
Highest rating, with at least 10 reviews	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2
Best value for money with at least 10 reviews and 4.5 rating	CANON IXUS 155 Compact Digital Camera - Silver	4.5 (111)	79.99	2

APPENDIX C – QUESTIONNAIRE SAMPLE

Following is a sample of the online questionnaire served in Typeform. The questionnaire was transferred as-is to Surviio, apart from question 12 that was omitted.







Have a look at our results (which you cannot skip!)

We have identified 3 possible usage scenarios for cameras and came up with different recommendations for each. You only need to view the results for the usage that seems more relevant to you. Please remember, we are always talking about COMPACT digital cameras.

http://goo.gl/0uLjDw - Typical everyday use
http://goo.gl/BsxZXQ - Shooting often at sports occasions
http://goo.gl/PPCd4F - Try to take more artistic photos

You can then come back, and continue this survey.

Continue press ENTER

1 → Before we get into details, in a scale of 1-4, how knowledgeable do you consider yourself about Compact Digital Cameras?*

A 1 - Just enough to take a shot
B 2
C 3
D 4 - Familiar with advanced camera features

2 → Awesome! For which usage scenario will you give feedback?*

A Typical everyday use
B Shooting often at sports occasions
C Try to take more artistic photos

3→ If you were to buy a camera, how likely are you to choose one of the recommended options.*

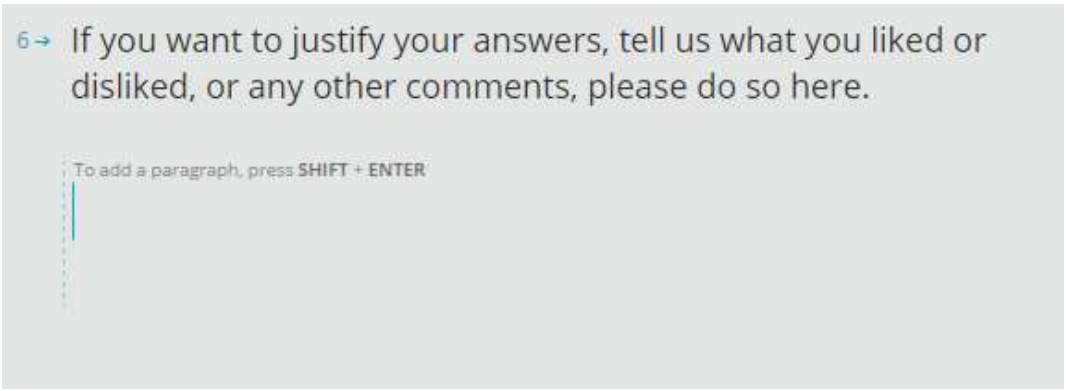
- A Very Unlikely
- B Unlikely
- C Neutral
- D Likely
- E Very Likely

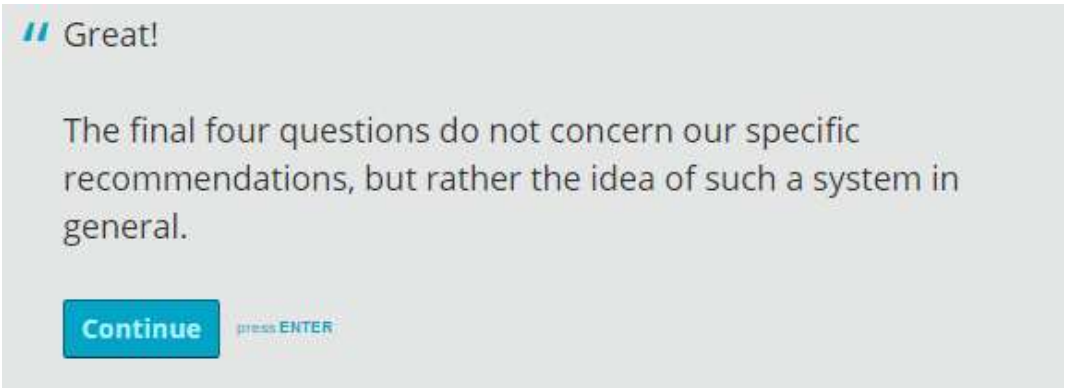
4→ How do you feel about the variation between the recommended cameras?*

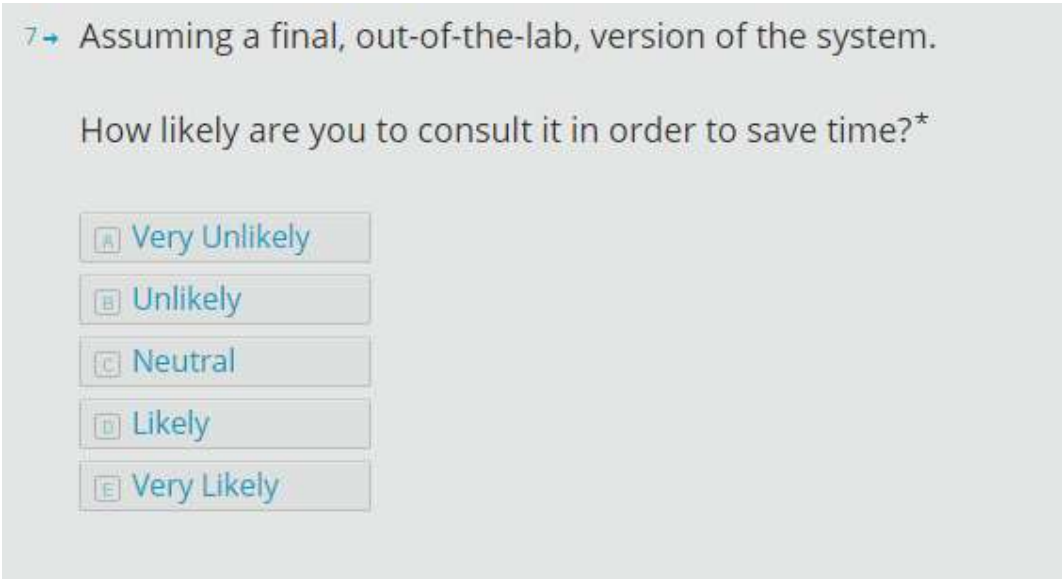
- A 1 - All cameras looked the same
- B 2
- C 3
- D 4 - Each camera offered something different

5→ In overall, how satisfied are you with our recommendations.*

- A Very Dissatisfied
- B Dissatisfied
- C Neither satisfied nor disatisfied
- D Satisfied
- E Very Satisfied


6 → If you want to justify your answers, tell us what you liked or disliked, or any other comments, please do so here.
To add a paragraph, press SHIFT + ENTER


Great!
The final four questions do not concern our specific recommendations, but rather the idea of such a system in general.
Continue
press ENTER


7 → Assuming a final, out-of-the-lab, version of the system.
How likely are you to consult it in order to save time?*
A Very Unlikely
B Unlikely
C Neutral
D Likely
E Very Likely

8 → Assuming a final, out-of-the-lab, version of the system.

How likely are you to consult it to find products better suited for you.*

- A Very Unlikely
- B Unlikely
- C Neutral
- D Likely
- E Very Likely

9 → In which of the following ways, would you be likely to use the system?*

Choose as many as you like

- A I would not use the system
- B To use the recommendations as a starting point for my market research
- C To buy one of the recommendations and skip doing my own research
- D Buy one of the recommended options, but only after validating it with another source.
- E To get an idea of the market
- F To find out what I should be looking for in a product, then do my own market research
- G Other

10 → At the moment, the system is searching for the following products to recommend

- Best value for money
- Best match
- Most popular
- Best price

Is there another recommendation that you would like to see?

11 → If you have any further comments, please leave them below.

To add a paragraph, press **SHIFT** + **ENTER**

12 → Oh before you go, have you answered questions for market troll before?*

Y Yes

N No

APPENDIX D – FORM UPR16

FORM UPR16

Research Ethics Review Checklist

University of Portsmouth

Please include this completed form as an appendix to your thesis (see the Postgraduate Research Student Handbook for more information

Postgraduate Research Student (PGRS) Information				**Student ID:**	435812		
PGRS Name:	Andreas Theodoros Lianos						
Department:	Engineering		**First Supervisor:**	Linda Yang			
Start Date: (or progression date for Prof Doc students)	**April 2012**						
Study Mode and Route:	Part-time	☒	MPhil	☐	MD	☐	
	Full-time	☐	PhD	☐	Professional Doctorate	☐	

Title of Thesis:	Artificial and Crowd Intelligence Based Recommender System Framework
Thesis Word Count: (excluding ancillary data)	37571

If you are unsure about any of the following, please contact the local representative on your Faculty Ethics Committee for advice. Please note that it is your responsibility to follow the University's Ethics Policy and any relevant University, academic or professional guidelines in the conduct of your study

Although the Ethics Committee may have given your study a favourable opinion, the final responsibility for the ethical conduct of this work lies with the researcher(s).

UKRIO Finished Research Checklist: (If you would like to know more about the checklist, please see your Faculty or Departmental Ethics Committee rep or see the online version of the full checklist at: http://www.ukrio.org/what-we-do/code-of-practice-for-research/)		
a)	Have all of your research and findings been reported accurately, honestly and within a reasonable time frame?	YES ☒ NO ☐
b)	Have all contributions to knowledge been acknowledged?	YES ☒ NO ☐
c)	Have you complied with all agreements relating to intellectual property, publication and authorship?	YES ☒ NO ☐
d)	Has your research data been retained in a secure and accessible form and will it remain so for the required duration?	YES ☒ NO ☐
e)	Does your research comply with all legal, ethical, and contractual requirements?	YES ☒ NO ☐

Candidate Statement:	
I have considered the ethical dimensions of the above named research project, and have successfully obtained the necessary ethical approval(s)	
Ethical review number(s) from Faculty Ethics Committee (or from NRES/SCREC):	8C18-9EB7-2787-28A1-BB5F-8E75-17B8-67C7 Further to the above fast-track certificate number, I attach the communication that confirms the acceptance. On 19 December 2013 15:54, Andy Mew <andy.mew@port.ac.uk> wrote: Dear John/Giles, Kazuya and Athena have confirmed they are happy with the ethical review form submitted by Andreas Lianos and agree that the student may undertake their research in the selected area. With John's approval included, this meets the three-person agreement required for a partial review. Many Thanks, Andy Andy Mew Research Officer Faculty of Technology 3rd Floor, Hippodrome House Guildhall Walk
If you have *not* submitted your work for ethical review, and/or you have answered 'No' to one or more of questions a) to e), please explain below why this is so:	
Signed *(PGRS)*: 	**Date:** 21/5/2016